1994

Multimedia in advanced telecommunications services

Vivek Beri
University of Wollongong

Abstract

This thesis investigates the multimedia in advanced telecommunications services. The advanced telecommunications services refer to the Universal Personal Telecommunication (UPT) services which are currently under research at the University of Wollongong.

A prototype of the user interface was developed incorporating UPT features that are relevant from a UPT user's point of view. Mobility in time, space, and media is discussed that allows a UPT user a high degree of personal freedom. Facsimile, email, video conferencing, and phone are modes of communication that are explored. Providing all of these modes on a user's terminal increases the user's communication capability drastically and give mobility in media. The UPT Diary enables the user to communicate in a time independent fashion.

Designing a good user interface is a difficult task. Good design principles that have been proposed by researchers in the field were taken into consideration at design stage. The user interfaces were later evaluated using some of the accepted design principles.

Acknowledgements

A number of people have contributed to the successful completion of this project.

For his guidance, I would like to thank my supervisor Professor Fergus O'Brien, for his constant support and encouragement throughout the course of this research. Every discussion with him left me full of new ideas to explore.

Acknowledgements are also due to Bernhard G. Humm for his guidance during this project and for helping me come to grips with the Smalltalk environment.

I would also like to thank Marilyn Cross for pointing me towards good material for my literature survey.

Last but not the least, I thank Gordon Cheng and David I. Raymond for their assistance in the preparation of this thesis.

Contents

Chapter 1

Introduction

1.1 Motivation

Universal Personal Telecommunication is the telecommunications service under research at TSRC[1], and it's main focus is on mobility in telecommunications. This is a new and emerging concept in the world of telecommunications and I found it to be a very fascinating one. The success of any new product or service ultimately depends on the acceptability of the device interface by the user. Designing user interfaces for such a service is a challenging and creative task. The challenge comes from providing a good user interface that minimizes the stress on the user while the user performs the task. The interface must be usable and not just be user friendly. The area of human-computer interaction covers a variety of fields such as, human psychology, anthropology, graphic design, computer science, cognitive science, and so on. It is not possible to study all of these areas in depth, but during the course of this research I had the opportunity

[1]Telecommunications Software Research Center at The University of Wollongong

to briefly touch upon these areas. I found the literature to be intellectually stimulating.

1.2 Objectives

The primary objective of this research was to examine the modes of communication that users currently use, and their potential combinations without concern as to the underlying network. In other words, to explore multimedia in telecommunication on the Universal Personal Telecommunication (UPT) user's terminal, and demonstrate the opportunities for the design of a UPT-terminal device.

The second objective was to provide user interfaces for the following four modes of communication to give the UPT users mobility in media.

1. Phone
2. Fax
3. Email
4. Video

Lastly, to provide an user interface to support the *Diary* concept to enable the UPT user to communicate in a time independent fashion enabling negotiation between the UPT caller and the called party.

1.3 Overview

The thesis has been structured into seven chapters. In the following paragraphs I have given a brief description of what each chapter is comprised. The first

chapter is Introduction and so I start with Chapter 2.

Chapter 2 deals with Human-Computer Interaction, and user-interface issues. A brief background of these topics is presented along with the design principles (Schneiderman's and Norman's) which I have used in my interfaces. The possibilities of the computer are limited not by its power to compute, but rather by its power to communicate with its human users. As interfaces become easier to use, they become harder to create. This chapter also has sections that discuss issues like ease of use, and design for simplicity among others.

Chapter 3 is an introduction to the Universal Personal Telecommunication service. UPT is a broad research area spreading into different topics such as distributed transaction processing, database and billing issues, security and privacy issues, network and hardware issues and so on. UPT terminology and services under UPT which are of interest to us have been defined in this chapter.

Chapter 4 defines multimedia and its scope under UPT. Concepts involving interface design, syntactic knowledge, semantic knowledge, and task concepts that helped design the multi-media user interface for UPT have been discussed.

Chapter 5 deals with the programming environment that was used to develop the prototype of the user interface. Smalltalk-80 has been introduced as being the developing environment. This chapter also discusses the salient features of Smalltalk-80, and the benefits of object-oriented design in developing a user interface.

Chapter 6 presents the design and implementation of the prototype of the UPT user interface. The different modes of communication—phone, facsimile, email, and video, that offer mobility in media are discussed and their interfaces presented. These interfaces are then evaluated on user interface design principles which were introduced in Chapter 2. Chapter 6 also explores the UPT *Diary*

concept which assists the UPT user to communicate in a time independent fashion, and presents a good user interface for it. Several illustrations depicting the actual user interfaces have been used to explain the design of interfaces.

Chapter 7 is the last chapter in this thesis. It talks about the conclusions of this research project.

Chapter 2

Human-Computer Interaction

2.1 Communication

"As we grow more familiar with the intelligent environment, and learn to converse with it from the time we leave the cradle, we will begin to use the computers with a grace and naturalness that is hard for us to imagine today. And they will help all of us—not just a few "super technocrats"—to think more deeply about ourselves and the world." [Toffler 1980]

We can observe first-hand how people learn to adapt to their purposes a new technology which allows exciting new possibilities for human beings to communicate thoughts across time and space via the computer. The key to the real effectiveness of computers is usability by people other than computer professionals. The possibilities of the computer are limited not by its power to compute, but rather by its power to communicate with its human users. Human-computer interaction is now an area of research and practice with broadly recognized impact and increasing rate of growth.

The way that we communicate today in face-to-face communications, via phone, writing, or computer, is largely determined by social habits built up over many thousands of years. In fact, our natural language capability was developed to allow communications under the constraints primarily imposed by our ability to produce sounds and secondarily by our abilities to write. With the advent of the printing press, film, television, and most importantly the computer, these limitations no longer dictate the structures of communications. Despite this potential freedom brought by computers, however, most communication between human beings still proceeds in a slow linear fashion: By the intelligent use of graphics and proper selection of pictures from the data banks of video images, entire arrays of data could be presented to users. We stand on the threshold of a new age brought about by new technology.

2.2 Creating User Interfaces

Building any software is a complex task, even when the system is computation intensive and does not emphasize the interface. Injecting the ever-changing, always-unsatisfied human into the process necessitates a quality human-computer interface, complicating an already complex task. There are no guidelines or techniques that guarantee the software will be easy to use, and software implementors have generally proven to be poor at providing interfaces that people like. It would be better if design teams involved people from different walks of life: computer scientists, graphic designers, psychologists, and the end users to design the interfaces [Tog92]. Consequently, interface software must often be prototyped and modified repeatedly.

As interfaces become easier to use, they become harder to create. The

easy-to-use, *direct-manipulation* (described in Section 2.7) interfaces are among the most difficult to implement. These interfaces let the user operate the objects that are visible on the screen, performing rapid, reversible, incremental actions.

An interactive system—one with human-computer interface—is not judged solely on its ability to compute. It is also judged on its ability to communicate. In fact, if users cannot communicate effectively with an interactive system, its computational ability may be inaccessible.

2.2.1 Ease of Use

A good user interface should be both easy to learn and easy to master. It should be reliable, efficient and "invisible". In addition to being emotionally appealing, fantasies that are analogous to things which the users are already familiar, can help make the systems easier to learn and use. The user interface for the Macintosh is an example of a system that makes extensive use of metaphors [AC87]. Much of the manipulation of information takes place by moving icons around on a "desk top" that is simulated on the screen. The icons are pictorial representations of familiar objects like file folders, trash bins, and filing cabinets. To the extent that this fantasy is analogous to desktops, it presumably makes the system easier to learn and use. However, must new technology be burdened with "metamorphic compatibility" constraints? The question here is, how can we present a scenario to new users to reap the benefit of prior knowledge and to avoid the pitfalls. People are accustomed to communicating with one another by means of natural language. Given a choice, many people would prefer to communicate with the computer in this fashion as well, since natural languages are well practiced, flexible, and powerful. With a natural language system the

user would not have to remember a large number of idiosyncratic command and procedure names, nor frequently refer to a manual, nor undergo specialized training. However, such systems are rare and expensive and it still remains an active research area. According to Dr Ben Schneiderman who is an advocate of the *direct manipulation* technique, AI (and the natural language approach) emphasizes on machines being smarter (or as smart) than the users and hence control the task. Whereas UI (User Interface) is all about designing systems that give the user more and more control over the machine. Supporters of natural languages and AI have yet to provide some evidence of their systems being very user friendly.

Many practical alternatives for user-computer interaction exist. All have advantages and disadvantages. At one extreme, the programming languages with precise syntax and vocabulary may be used for dialog between person and computer. The languages have the advantage that complex concepts can be communicated unambiguously by the user. The drawback is that the use of the language requires extensive training and strict adherence to rules. At the other extreme, a hierarchy of detailed menus may be used for person-computer interaction. Since this technique relies heavily on the user's recognition memory and passive response to computer prompts, little formal training is required. The drawback is that the creator of the menus must have the perception of all the desirable options to include. Since the choice cannot be infinite, limitations in the actions through the use of menus is unavoidable. Furthermore, the use of the menus may be tedious if the choices are too finely detailed or if the user has extensive experience. It would be a good idea to provide shortcuts and layers/levels for expert users to quickly get around their task. Somewhere in between these extremes lie techniques such as the form-filling or fill-in-the-blanks.

Researches have attempted and succeeded in developing user interfaces that satisfy a broad spectrum of users, from competent system programmers to complete novices [TS85]. This is of special interest to me as the users of advanced telecommunications services would be of several levels with a different degree of expertise or experience in computer usage.

2.3 Designing a User Interface

Designing an effective user interface to a complex information system is difficult, since it cannot be done in isolation from the design of the information system itself. For example, an office-system provides a variety of methods of composing and storing documents and transmitting them between individuals. Similarly, subscribers (users) of the UPT[1] (the advanced telecommunications services refer to UPT, the application domain) will need to communicate with other users and other individuals via telecommunications. At the abstract level of description we are not concerned with the exact structure of documents, the underlying network, and the organizational structure (i.e., the authority/responsibility relationships between people). My main concern is to develop a prototype of the user interface to the advanced telecommunications services.

2.3.1 Design Process

The place of human factors in relation to various stages of the design process, and the best procedures for assisting the designers to achieve good usability design, have been studied empirically for many years. From these studies, one

[1]Universal Personal Telecommunications

can synthesize and propose a set of fundamental features which will probably find widespread acceptance by experienced human factors specialists as key precepts for the process of design for usability. The five fundamental features of design for usability according to B. Shackel are [Sha85],

- user centered design
- participative design
- experimental design
- iterative design
- and user supportive design.

User-Centered Design

Designers must understand who the users will be and what tasks they will be doing. This involves user designer interaction, and the designer must focus from the start on users and tasks.

Participative Design

A group of expected users should work closely with the designers, especially during the early formulation stages and especially when creating the usability specification. To enable the users to make useful contributions, they will need to be shown a range of possibilities and alternatives by means of mock ups and simulations. The users feedback may reveal potential problems that should be considered before they are embedded into the design.

Experimental Design

Early in the development process the expected user should do pilot trials and then subsequently use the simulations, and later the prototypes, to do the real work. Whenever possible alternative versions of important features and interfaces should be simulated or prototyped for evaluation by comparative testing. These studies should be formal, with measures of the performance and the subjective reaction of the users. Thus, ease of learning and ease of use can be assessed and the difficulties removed.

Iterative Design

The difficulties revealed in user tests must be remedied by re-design, so the cycle—design, test and re-design must be repeated as often as is necessary until the usability specification is satisfied.

User Supportive Design

This area is often left until a very late stage in the design process, and then some documentation and help screens are written in a hurry at the last minute. Careful attention to training, selection manuals, quick reference cards, "help" systems like on-line context sensitive help and off-line help can significantly assist usability.

2.4 Need for Good User Interface

Computers today are used for a very broad range of applications, and their use is increasing every day. Some computers may be embedded in larger systems, so

that they communicate only with other computers and not directly with human users. In such a case there is no user interface needed. Some computers are designed as general tools which can be adapted by a skilled users for whatever purpose they desire. A user must provide exact instructions to program the computer to perform any task at hand. Over the decades computers are being used by more and more of not so skilled users as well. Different applications run on computer systems and even the most novice of users interacts with the machine and gets his/her work done through sophisticated user interfaces. As mentioned in Section 2.2.1, the users are varied, also shown below, and careful design is needed in developing a good user interface.

- Information systems support human *users performing defined tasks* . Careful design of the user-interface system will be needed to ensure effective system operation.

- Users differ in ability, training and job experience. They may be keenly concerned with task performance, but may have little knowledge of computers themselves.

2.5 User-System Interface

What is user-system interface? In common usage, the phrase is broadly defined to "include all aspects of system design that affect system use" [Smi82]. It is obvious that software is not the only significant factor influencing user performance. Other aspects of human computer interaction are clearly important, including users ability and experience, workstation design, keyboard layout, physical display characteristics, environmental factors such as illumination and

noise, etc. To achieve a good user interface design, all of those factors must be designed with care. Accommodating the diverse human perceptual abilities is a challenge to every designer. Many studies have been conducted in the past to design (statistically) peripherals to facilitate computer use. Designers need to be aware of the ranges of human perceptual abilities. Ben Schneiderman [Sch87] has pointed out a list of concerns about perception which includes parameters such as visual fatigue, flicker sensitivity, etc. Other senses such as touch (keyboard, touchscreen, etc.) and hearing are also important. These physical abilities influence the design of the workstation (or playstation).

However, most of these design considerations are hardware related and our main concern here is the multimedia on the users terminal. It would not be wrong to assume that workstations designed to these concerns are already available, and it is for us to provide a sophisticated software user interface. The physical design of workplaces is often discussed under the ergonomics. anthropometry, sociology, industrial psychology, and organizational behavior. They have been listed here to highlight their importance in human-computer interaction.

2.6 Design Process

There is now a convincing evidence that creative design is not at all linear, one-step-at-a-time; it is neither top-down nor bottom-up. Design is better described as a search than as a computation [Row87]. Designers actually jump around, following semantic links, and constraint relations in the design space. This process has been referred to as *opportunistic design*. Dr Donald A. Norman [Nor88b] who has done a lot of work in psychology. has dealt with the issues

of *constraints* and *affordances*. The *Action Cycle* which he proposed, offers designers a good guideline while designing the user interface. In the following subsection we talk about Norman's Action Cycle, and then use it to evaluate our user interface design in a later chapter. Also, I have used principles of *direct manipulation* to evaluate the UPT user interface prototype and have described the concept in the following section.

2.7 Direct Manipulation

A direct manipulation user interface presents users with a model of their information space and they modify their information by direct action (explicit commands are not necessary). Such interfaces generate an enthusiasm among users. The advantages of direct manipulation are:

- Users feel in control of the computer—mastery of the system.
- Typical user learning time is short and new advanced features are easily assimilated. Novices learn basic functionality quickly, usually through a demonstration by an experienced user.
- Users get immediate feedback on their actions so slips can be detected and corrected quickly.
- Users enjoy using the system, and are eager to show it off to novices.
- Error messages are rarely needed.

The reasons for why it is pleasing for an user to use such a system revolve around the following:

- The objects and actions of interest are visible.
- Actions are rapid and reversible.
- Complex command language syntax is replaced by direct manipulation of objects.

The principles of direct manipulation have been described by several authors. "What you see is what you get", described by Don Hatfield [Hat81]. This was expanded to, "What you see is what you have got" [Thi82]. It was suggested by Thimbleby that the display should indicate a more complete image of what the status is. The concepts of direct manipulation were reviewed by Hutchins et al. and they describe it as the "feeling of involvement directly with a world of objects rather that of communicating with an intermediary", [HHJ+86].

As computer systems are increasingly used by casual and untrained users, direct manipulation interfaces will become more common. Dealing with representations of objects is more "natural" and closer to innate human capabilities: action and visual skills developed well before language in human evolution.

2.8 Design For Simplicity

For the user (of any system) to accomplish any task, the goal(s) must be satisfied. Unless and until the desired goal state is achieved the task is incomplete. The user gets the feedback from the world and uses this feedback to get closer to the goal. A goal maybe divided into subgoals and all these subgoals must be satisfied in order to accomplish the original goal. In artificial intelligence we have dealt with problems which generate large and complex goal trees. For example, the common chess problem, or the traveling salesman problem, or the

the cannibal and missionary problem. However, in real life the everyday tasks (problems) are a lot simpler. For example, choosing from a menu in a restaurant is fairly easy. The menu may have a lot of different preparations to offer (wide tree structure) but once you have decided on what you'll like to eat then that's it, the search is over. Everyday tasks have to be shallow and wide or deep and narrow because:

- They must be performed quickly;
- without much mental effort.

The advanced telecommunications services may be classified as everyday tasks and hence their design must be shallow and narrow.

2.8.1 Action Cycle

The Action Cycle proposed by Dr Norman is illustrated in Figure 2.1.

The Action Cycle starting from the *goal* consists of two phases. The first phase is the *Execution* phase which is doing something. The second phase is the *Evaluation* phase which is the comparison of what happened with what we wanted to happen. Each of these two phases consist of stages which are listed as follows.

1. **Execution Phase**

   Forming the Goal: Goal is what we want to happen, that is the state that we want to achieve.

   Forming the Intention: The goal is translated into an *intention* to do some action.

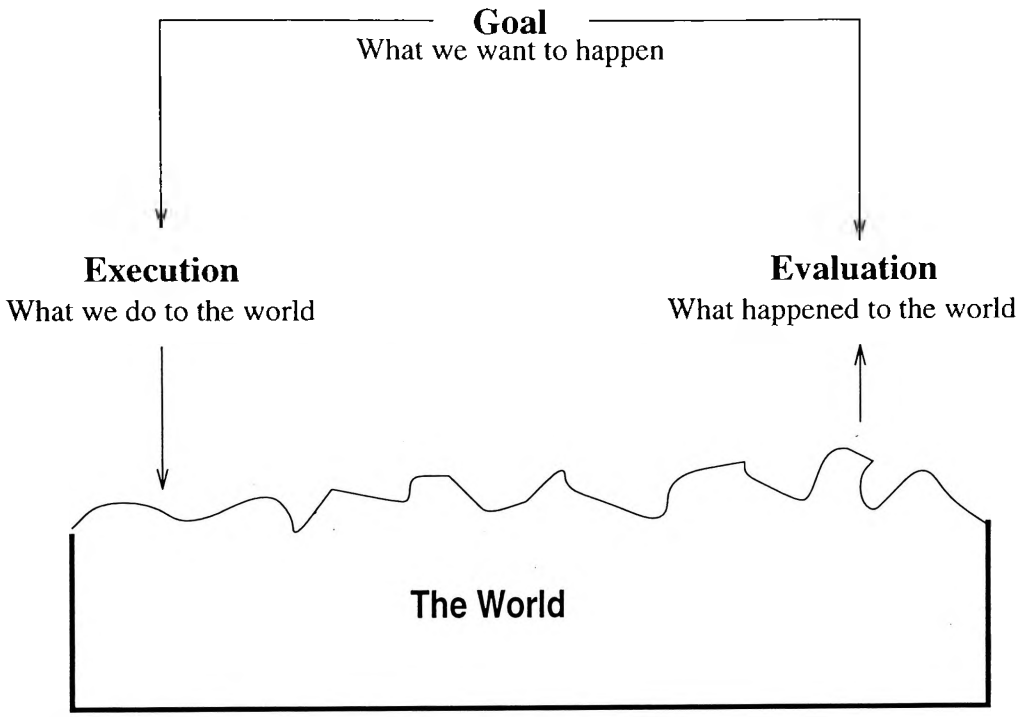

Figure 2.1: The Action Cycle.

Specifying an Action: The intention is translated into a set of internal commands or an action sequence that can be performed to satisfy the intention.

Executing the Action: The above stages are all parts of the mental model. No change occurs unless the *action sequence* is *executed*.

2. **Evaluation Phase**

Perceiving state of the world: Evaluation starts with our *perception* of the world.

Interpretation: This *perception* of the world is then *interpreted*.

Evaluation: The *interpretation* is then *evaluated* with respect to the intentions and the goal.

It is important to know what these stages are and how they should be used. On top of that there is a trade-off among them. Making one easy may make the other stages harder. For example, in the case of the automatic pilot eject button in a combat aircraft, the button is hidden behind some panel and is kept out of the way from other controls. This is designed to avoid the eject button from being pressed accidently. The design goal here is to avoid the pilot from ejecting into the sky accidently, but letting him/her eject out in case of an emergency. However, if the eject button is moved onto the front panel then it's execution (pressing the button) is simplified but the chances of pressing it accidently have also increased.

2.8.2 Questions at Design Time

We can use these seven stages in the design process. Taking the stages one at a time we could ask ourselves questions at design time. The questions for each stage are relatively simple, and these in turn, boil down to the principles of good design. We could ask the following questions for each of the seven stages.

Goal Stage: How easily can the user determine the function of the device?

Intention Stage: How easily can the user tell what actions are possible?

Action Specification Stage: How easily can the user determine the mapping from intention to physical movement?

Execution Stage: How easily can the user perform the action?

Perception Stage: How easily can the user tell what state the system is in?

Interpretation Stage: How easily can the user determine the mapping from system state to interpretation?

Evaluation Stage: How easily can the user tell if the system is in the desired state?

To summarize, the seven design principles are as follows.

1. Use both knowledge in the world and in the head.
2. Simplify the structure of tasks—not wide and deep.
3. Make things visible.
4. Get the mappings right.
5. Exploit the constraints.
6. Design for errors.
7. When all else fails—standardize.

The above mentioned design questions do not quantify the usability of the design but provide important guidelines to the designer at design time. The designer is made aware of these issues that he/she can keep in mind while proceeding with the design.

2.9 Raising the Consciousness of the General Public

With the rapid advances in technology, we are attempting to provide very powerful facilities and tools in the hands of novices. A vast majority of human

population (from industrialized nations) is quite comfortable with a bank terminal or word processor. The initial fear of the machine "being smarter than I am" has now been overcome at least in everyday applications. People are getting more and more used to working and living with computers and are accepting them. Still, there are people who may be fearful of making mistakes, damaging equipment, worried about feeling incompetent, or threatened by the computer. These fears are generated in part by poor designs that have complex commands and vague error messages. As examples of successful and satisfying designs become more visible, the crude designs will appear increasingly archaic and become commercial failures. Hence, it is very important that we provide user interfaces which will be easy and convenient to use and also provide all the facilit ies and functionalities.

2.10 User Interface Software

The function of each button, the functional arrangement of buttons, the size and distribution of elements within a display are established not in the design of the equipment but in how the computer is programmed—the software. The "design" in the programs establishes the contents of processed data available to the operator and the visual relationships among the data. Operator actions and the feedback that he/she gets can be established. Continuing concern for user interface software is suggested by phrases such as "software psychology" (cf. Schneiderman, 1980). However, user interface design cannot be the concern only of the psychologist or the human factors specialist. It is a significant part of information system design that must engage the attention of system developers, designers, and ultimately system users as well.

Studies conducted around the mid eighties [SM84] estimated the user interface design to comprise of 30 to 35 percent of operational software. Estimates of individual systems ranged from 3 to 100 percent, reflecting the fact that some computer systems require a much higher investment in user interface design than others, depending upon their purpose.

The design of user interface is not only expensive and time consuming, but it is also critical for effective system performance. Users in general can compensate for poor design with extra effort. Probably no single user interface bug, in itself, will cause system failure. But there is a limit to how well the users can adapt to poorly designed interface. As one deficiency is added to another, the cumulative negative effects may eventually result in system failure, poor performance, and/or user complaints.

2.10.1 Data Entry

Data entry refers to user actions involving input of data to a computer, and computer responses to each inputs. The simplest kind of data entry consists merely of pointing at something—selecting an item or designating a position on a computer-generated display. This is also the most natural way for the users to respond, pointing and selecting—direct manipulation. The computer will also play a role in the data entry process, guiding users who need help, checking data entries to detect errors, and providing other kinds of data processing aids. As designers of user interface, we must be concerned about computer processing logic as well as data input by users.

Data can be entered into a computer in a variety of different ways. Users might designate position or direction by pointing at a display. Users might enter

numbers, letters, or more extended textual material by keyed inputs, or in some applications by spoken inputs. Data might be keyed into displayed forms or tables, into constrained message formats, or as free text. In graphic interaction users might draw pictures or manipulate displayed graphic elements.

2.10.2 Data Display

Data display refers to computer output of data to user, and assimilation of information from such outputs. Some kind of display output is needed for all kinds of information handling tasks.

2.11 Usefulness of Object Orientation in HCI

Human-computer interface development tools need the ability to support rapid design changes without recompiling or relinking, which can take substantial amounts of time for large applications. When prototyping, it is desirable to see the effects of changes quickly in order to proceed with the design. Some object-oriented programming languages offer this advantage over conventional programming languages. Also, because of its event based nature, object orientation is effective for representing asynchronous dialogue and for representing the behavior of specific interface features (e.g., windows) regardless of their context. Object-Orientation and Rapid Prototyping will be discussed in the forth coming chapters.

Chapter 3

The Application Domain: Universal Personal Telecommunications

3.1 UPT Service

In telecommunications, the term "service" means a complete set of capabilities provided by networks and offered to users (for example end-users, network providers and service providers) [HF92]. Examples for telecommunications services are Freephone (for end-users), X.500 electronic diary (for service providers), and X.400 message handling (for network providers).

UPT[1] is the telecommunications service under research at TSRC[2] at the University of Wollongong. It's main focus is on the *mobility* in telecommuni-

[1]Universal Personal Telecommunications

[2]Telecommunications Software Research Center

cations.

The CCITT draft recommendation F.851 [CCI] defines UPT as:

"UPT enables access to telecommunications services while allowing personal mobility. It enables each UPT user to participate in a user-defined set of subscribed services and to initiate and receive calls on the basis of a unique, personal, network-independent UPT number across multiple networks at any terminal, fixed or mobile, irrespective of geographic location, limited only by terminal and network capabilities and restrictions imposed by the network provider".

3.2 Telecommunications Services and UPT

Universal Personal Telecommunication (UPT) extends to the UPT user the capability to participate in a prearranged set of telecommunication services irrespective of geographic location. This is achieved by removing the relationship of the user identity from the terminal identity, thus providing personal mobility across multiple networks. In fixed telecommunication networks, users are associated with the network access of the terminal (i.e. the identification). In current mobile telecommunication networks, users are associated with the terminal in use (i.e. terminal identification).

For UPT, this association is moved a step further away from the network. The UPT user is associated with a personal UPT Number. In order to offer users the capability of establishing and receiving calls on any terminal and at any location, the identification of users must be treated separately from addressing of terminals and network access. The user does not need to know the location of any stored information associated with UPT numbers (e.g. service profiles) in order to use UPT [CCI].

UPT replaces the traditional approach to accessing people via telephone numbers (device addresses) with accessing people via personal UPT numbers (logical addresses). In order to provide the service to the end-users, the Telecom engineers face challenges in meeting end-user needs. It is the acceptability of the interface device to a user that ultimately determines the success, or at least partial success, of any service [O'B91]. UPT facilities should be easy-to-use and consistent across terminal types and geographic and network boundaries so as not to deter the user from using UPT. A good design issue would be to consider the user not entering (manually) a large amount of information for basic features.

3.3 Services Under UPT

The implementation of UPT on a world-wide basis will not only have major impacts on the underlying networks but also on the way end-users use telecommunications services. The users will have access to multimedia communication through elegant user interfaces. The media are listed as follows:

- phone
- electronic mail
- facsimile
- and video conferencing.

The UPT services will provide communication in all of the above listed media, as well as provide additional user flexibility. The *diary* concept is an example of

such flexibility. This facility enables the UPT user to preset the mode of communication (facsimile, email, phone etc.) for a specific interval of time. During this interval the UPT user will receive all communication in his/her specified mode. The workstations or terminals that the user would use to communicate will need to have interfaces to microphones, video cameras, fax machines etc. Providing a good user interface will enable the user to control all such devices from his/her terminal.

In the prototype developed, interfaces are provided for each of the above mentioned media, and are discussed in the chapter of Design and Implementation. Here, the intention was only to introduce these different media as parts of the UPT user interface.

Chapter 4

A Multi-Media Interface for UPT

4.1 What is Multimedia?

Multimedia enables people to communicate using integrated media: audio, video, text, graphics, fax, and telephony. The benefit is more powerful communication. The combination of several media often provides richer, more effective communication of information or ideas than a single media such as traditional text-based or voice-based communication can accomplish.

4.1.1 The Network and Platform for Powerful Communication

Powerful networking capabilities provide access to people and information across not only the enterprise, but cross-platform and globally as well. People and information must also be accessible when they are not at their desk through

portable computing and remote access. For instance, you could call your workstation from a public phone and have important information read to you. Such facilities would be available if the UPT had been implemented.

The networked multimedia desktop is a prerequisite for collaborative multimedia; powerful communication depends on a powerful platform. This platform must be multimedia-ready, with bundled multimedia technology. Connectivity (networking), speed (workstation performance), and multi-tasking are all necessary to enable this level of communication power.

Most of these issues are outside the scope of this thesis and form a part of the extensive research being done on UPT at TSRC. We are only concerned with the user-interface for multimedia on the users terminal.

4.2 Scope of Multimedia Under UPT

"Imagine a classroom with a window on all the worlds knowledge. Imagine a teacher with a capacity to bring to life any image, any sound, any event. Imagine a student with a power to visit any place on earth at any time in history. Imagine a screen that can display in vivid color the inner workings of a cell, the births and deaths of stars, the clashes of armies, and the triumphs of art. And then imagine that you have access to all of this and more by exerting little more effort than simply asking that it appear—John Sculley." [AH88]

After reading the above lines one gets an idea of the kind of information that would be accessible with the multimedia and hypermedia technology. The whole process of communication will be transformed, and powerful tools of communication would be placed in the hands of subscribed users. As the gap between computer technology and telecommunications decreases day by day, communi-

cations is getting computer based and very soon with the new networks (fibre optic and ISDN) complete, the computer terminal or workstation will replace the traditional phone instrument. Users of these advanced telecommunications services will engage in multimedia communication.

For instance, a UPT user calls a friend overseas on the phone. The friend is not at his/her desk and is unable to receive the call. Under current telecommunications services our user will have to try again later on. The user might decide to send a fax message to the friend overseas. In that case the user will have to go where the fax machine is, and that might mean going into another building. All this hassle will be removed under the UPT service. Providing multimedia on the users terminal will enable the UPT user to send fax or email or whatever he/she is subscribed to without having to move from the chair. With a few mouse clicks the appropriate interface will appear, allowing the user to communicate with his/her friend.

In the sections of Chapter 6 I have discussed the various media that were explored.

4.3 Interface Design

It has been standard practice until recently to write software without considering human capabilities and the human environment in which the software would be used. This strategy offered poor usability but sufficed for two reasons [TK89]:

- People are adaptable and they eventually learn how to deal with unfriendly software.

- Designing systems without any regard to human factors was a norm, there was little competitive pressure to build software on the basis on how people really are.

Many characteristics can influence user's performance on a system. They include age, attitude toward technology, prior experience with computers, job satisfaction, and curiosity. In telecommunications services the user domain is universal. In order to bridge the user gaps (caused by individual differences and motivation) being aware of the kinds of gaps that exist may have some utility for designers and developers. While multiple techniques may be used to address particular ecological gaps, we present techniques in terms of where they can be best applied in the cycle system design. The development cycle is as follows.

- Analysis phase to determine user requirements.
- The next phase is an iterative process, alternating between design, implementation of the prototype, and testing. This phase is affected by the skills and intuition of the designers, applications of human factors guidelines, testing, and by design principles.

4.3.1 Analysis Phase

Understanding user requirements is critical to creating a usable system. An effective technique before design begins is to start with observations in the target environment. Work has been done in this area by researchers and social, computing-environment, physical, and motivational issues have been proposed to be worth looking into. The analyst must try to understand the range of user, task, artifact, and work contexts that the system both will and could be used in.

4.3.2 Alternating Phase

Perhaps the most important recommendation during design is to use good software-development principles. For example, if the code is rapidly modifiable, the results of usability testing during iteration can be more easily applied. In addition, usability principles and guidelines (as discussed in Chapter 2) are used to increase the chances of better usability in the real world. In our system life cycle the second phase involves design, implementation, and testing. It is a cycle in itself.

Interface design requires creative leaps, and individuals vary tremendously in their ability here. One should aim towards maximizing the influence and capability of good designers. Instead of reducing the variance of design quality by forcing all designers to use a standard method, an alternative is to amplify and support their creative abilities by providing design aids.

In many contemporary systems, there is a grand opportunity to improve the human interface. The cluttered and multiple displays, complex and tedious procedures, inadequate command languages, inconsistent sequences of actions, and insufficient informative feedback can generate debilitating stress and anxiety. This leads to poor performance, frequent minor and occasional errors, and job dissatisfaction.

4.4 Syntactic Knowledge

When using a computer system, users must maintain a profusion of device-dependent details in their human memory. These low-level syntactic details include the knowledge of which key erases a character (delete, backspace) and

which icon to click on to scroll text forward, and so on.

The learning, use, and retention of this knowledge is hampered by two problems [Sch87]. First, these details vary across systems in an unpredictable manner. Second, acquiring syntactic knowledge is often a struggle because arbitrariness of these minor design features greatly reduces the effectiveness of learning. Rote memorization requires repeated rehearsals to reach a competent level, and retention over time is poor unless knowledge is frequently applied. Formal notations are useful for knowledgeable computer scientists but usually confusing to most users.

Syntactic knowledge is arbitrary, system dependent, and ill-structured. It must be acquired by rote memorization and repetition. Unless it is regularly used, it fades away from memory.

4.5 Semantic Knowledge

Semantic knowledge in human long-term memory has two components [Sch87]: computer concepts and task concepts. This accommodates the two most common forms of expertness:

- task experts who may be computer novices;
- computer experts who may be new to the task.

Computer concepts include objects and actions at high and low levels. For example, a set of computer concept deals with storage. At a high-level, for the users the concept is that computers store information. The concept of stored information could be refined into the object concept. Directory and files of information are separate objects. In turn the directory object is refined into

directory entries, and each entry that has a name, date of creation and so on. The file object may have a lower level structure consisting of lines, characters, and so on.

The computer actions are also decomposable into lower actions. Considering the same example, actions or goals would have different levels. The high-level goal, such as creating a text data file, may require load, write, and save actions. The mid-level action may be refined into storing a file and a backup file on a disk, of assigning a name to a file, and so on. At the low-level there are details about file types or sizes, storage space, and so on. Finally, there is the low-level action of issuing a specific command.

Since semantic knowledge about computer concepts has a logical structure and since it can be anchored to familiar concepts, this knowledge is expected to be relatively stable in human memory. If one remembers the high-level concept of saving a file, one will be able to conclude that the file must have a name, and a storage location, and so on.

4.6 Task Concepts

The primary way for people to deal with large and complex problems is to decompose them into smaller problems until each subproblem is manageable. Similarly, task actions can be decomposed into smaller actions. Making a phone call from a public telephone booth can be reduced to a series of steps: pick-up the handset, insert coins or phonecard, and press the appropriate number buttons. In sending an email message with a computer, the user has to integrate smoothly the three forms of knowledge. The user must have the high-level concept of writing (task action) a message (task object), recognize that the message could

be stored as a file (computer object), and know the details of commands such as the SEND command (computer action and syntactic knowledge). While writing a message it is expected that the user is fluent with the concept of composing a sentence, must recognize the mechanism for beginning, and ending a sentence, and finally, must know the low-level details of spelling each word, and know which ke y to press in order to compose the message.

Integrating the three forms of knowledge, the objects and actions, and the multiple levels of semantic knowledge is a substantial challenge that takes great motivation and concentration.

4.7 What All This Means to Us?

The knowledge of the concepts discussed in the above sections (Interface Design, Syntactic Knowledge, Semantic Knowledge, and Task Concepts) has been used in designing the multi-media user interface for UPT. Realizing the drawbacks of user interfaces which required syntactic knowledge and caused frustration and stress to the users, I decided to design the interface based on users semantic knowledge and task concepts. It has been shown that users are more at ease with the latter interfaces as they are able to make associations fairly easily within the application environment. Icons and screen layouts that are extensively used by Sun Microsystem Software, and the Macintosh, were incorporated to reduce variability and hence conform to some of the standards which are gaining acceptance.

Chapter 5

Object-Oriented Programming and Smalltalk-80

5.1 Background

Currently there are over two thousand different high-order programming languages. We see so many different programming languages because each was shaped by the particular requirement of its perceived problem domain. The existence of each new language enabled developers to move on to more complex problems. Along the road designers learned new lessons that changed their basic assumptions about what was important in a programming language and what was not.

The most recent advances in programming languages have been due to the influence of the object model. There are over a hundred different object-based and object-oriented programming languages today [Boo90]. The use of object-oriented design is not restricted to any one particular language and it is ap-

plicable to a wide spectrum of object-oriented and object-based programming languages. As important as design is, we cannot ignore the importance of the choice of the programming language, for ultimately our software must be expressed in some language. As suggested by W. Wulf, a programming language serves three purposes:

- "It is a design tool,
- It is a vehicle for human consumption,
- It is a vehicle for instructing a computer" [Wul80]

We chose Smalltalk-80 for reasons which are outlined in the succeeding sections.

5.2 Object Orientation

Software engineering practice shows that an object-oriented decomposition is generally more stable and more robust than a procedural decomposition [CY91]. Object-oriented decomposition yields smaller systems through the reuse of common mechanisms, thus providing an important economy of expression. Object-oriented systems are more resilient to change and thus better able to evolve over time, because their design is based upon stable intermediate forms. There is a seamless transition from the designer's model of the design work to the human interface design itself. In addition, object-oriented techniques are becoming well known in the software engineering community. So an object-oriented model tends to be acceptable, understandable, and attractive to engineers.

The characteristics of object-oriented programming—encapsulation (data abstraction and information hiding), dynamic binding, inheritance of attributes and functions, and message passing—make it a likely platform for direct-manipulation dialogue and asynchronous control. The advantages of object-oriented programming include higher productivity because code can be reused.

Object-oriented concepts have been touched upon very briefly as follows.

Class A class is an abstract data type that defines an interface with an implementation that supports instantiations of the class. A class can be seen as a user-defined data type with data as characteristics and methods as a fixed set of operations that characterize its behavior.

Abstraction Abstraction lets OOP programmer's encapsulate data and operations (methods) to form abstract data types.

Encapsulation The concept of encapsulation is preventing the viewing of the inside of the class. Encapsulation and abstraction are complementary concepts, where abstraction deals with the outside view of a class. In general encapsulation may be defined as hiding information.

Inheritance Inheritance is the concept that allows OOP programmers to reuse and extend existing (and presumably bug-free) classes. Thus inheritance improves the reliability of programs and reduces program development time.

Modularity Partitioning a program into individual components reduces the complexity of the program. These individual components or modules are handled differently in different languages. For instance in Smalltalk a class is the physical unit of decomposition, whereas in languages such as CLOS

and Object Pascal the module has software constructs. The modules are logically connected to each other and should have minimum dependencies.

Class hierarchy A class hierarchy is nothing but reusable code. None of it is specific to one application, but for that fraction which is used to directly implement the language and custom environmental features.

5.3 Rapid Prototyping and User Interface

What is rapid prototyping? Mark Mullin [Mul90] has described rapid prototyping to be a process where specification for a piece of software are developed by interaction between a software developer, a client, and a prototype program. Rapid prototyping is used when a client cannot initially define the requirements for a piece of software to a degree necessary to satisfy more traditional design methodologies.

Rapid prototyping is not just a "feel good" process. It operates within a clearly defined paradigm, expects certain environmental constraints to be satisfied, and expects developers to proceed from start to finish by way of a specific path. What is different about rapid prototyping is that it does not require as much initial definition, but allows for the definition of the problem to be developed over time.

Prototyping replaces anticipation (how will the system behave with analysis, and how do es it actually behave?) which is much easier to work wi

User interface prototyping is an essential part of the project and, unlike the prototyping of system functionality, it is sometimes acceptable to present an interface prototype as a system specification. Indeed, because of the dynamic

nature of user interfaces, paper specifications are not good enough for expressing the user interface requirements.

5.4 Code Reuse and User Interface Design

Only after you have learned to read other people's code and become familiar with what is available to you in the class hierarchy (described in Section 5.2), can you begin to create reusable code. Reusable code is one of the most powerful concepts in the object-oriented language paradigm, for it means that once you have solved a particular problem you never need to solve the same problem again. You simply reuse your original solution.

The creation of reusable code requires a deeper understanding of the nature of code so that you can separate the general from the specific. Usually we are trying to solve some application-specific problem and are attempting to get as much general purpose code as we can from the solution of the applications requirements. This is indeed the crux of code reuse, the fact that reusable or generic code is being produced in the process of developing a specific application.

5.5 Smalltalk-80

Smalltalk was created by members of the Xerox Palo Alto Research Center Learning Research Group as a software element of the Dynabook[1]. Simula was the major influence over Smalltalk, but Smalltalk took some ideas from the language FLEX. The work of Papert and Feurzeig also had its influence on the development of Smalltalk.

[1]a project of Alan Kay at Xerox PARC

Smalltalk represents both a programming language as well as a software development environment. Smalltalk is probably the most important object-oriented language, because its concepts have influenced not only the design of every subsequent object-oriented programming language, but also the look and feel of graphic user interfaces such as Motif and the Macintosh user interface, which have found large acceptance.

Ingalls states that "the purpose of the Smalltalk project is to support children of all ages in the world of information. The challenge is to identify and harness metaphors of sufficient simplicity and power to allow a single person to have access to, and creative control over, information which ranges from number and text to sounds and images" [Ing].

The Smalltalk programming environment includes a graphical user interface. Most system interaction is via menus where selections are made by pointing with the mouse. In Smalltalk, classes and objects implement event handlers. Event handlers are sent to objects as messages. Window management, for example, is accomplished by instances of window objects. Because all knowledge of window operations resides in classes such as *View* and *ScheduledWindow*, you simply send a message to a window object and it performs the required window operation on itself.

Smalltalk [GR89] is a good prototyping language for two reasons:

- The object-oriented nature of the language makes the system resilient to change.
- A large number of components are available to the programmer which may be incorporated in the prototype under development.

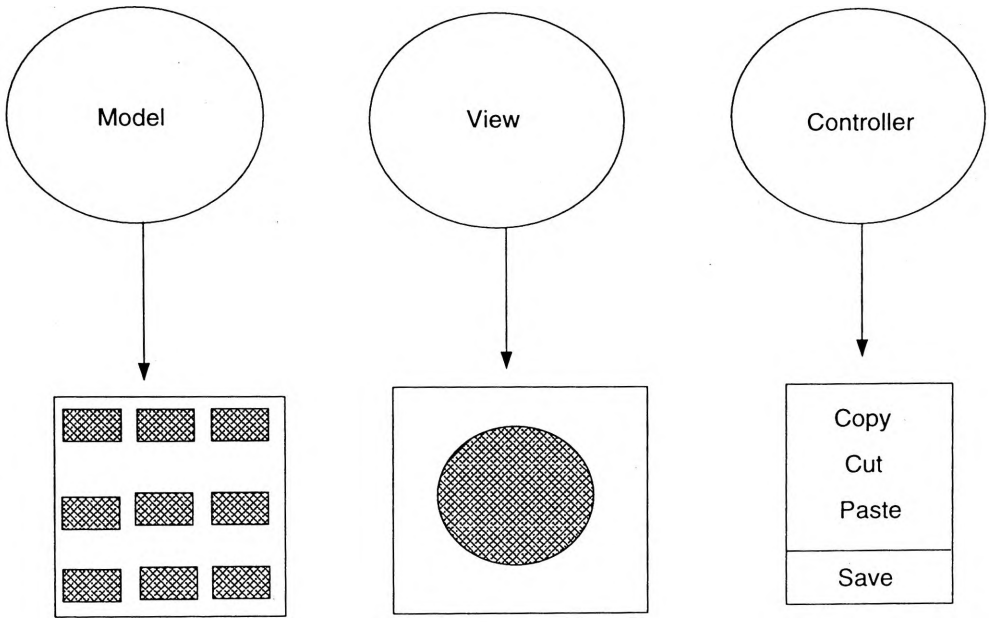

Figure 5.1: The MVC Architecture.

5.6 MVC Architecture of Smalltalk

Objectworks/Smalltalk provides the programmer an architecture to *model* , *view*, and *control* (MVC) the application. A model object provides the storage and processing of data, a view corresponds to the output, and a controller manages the user input. The Objectworks/Smalltalk Release 4 User's Guide illustrates model, view, and controller with their real-world counterparts. This is depicted in Fig. 5.1.

Researchers in HCI agree that there is a need to separate user interface from the data structure in order to yield good user-interface software. The MVC architecture allows programmers to produce object-oriented design in two stages, beginning with the model. The controller and view aspects are dependent on the model. The model has to know about its views and controllers, or else a change in the data cannot be reflected in the appropriate places on the display

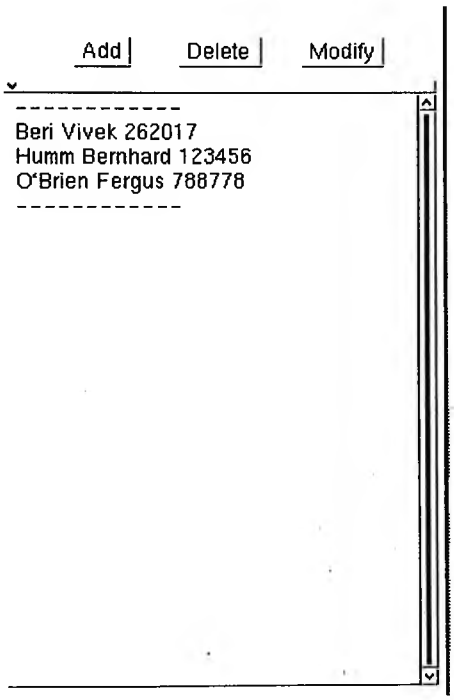

Figure 5.2: Scrollable List

and in the menus. A dependency mechanism is provided by the Object class (an Object class is the root class in the Smalltalk class hierarchy). Each dependent view adds itself to the model's list of dependents. Whenever the model changes in some way, it broadcasts a message to all dependents and they then take appropriate action. The result being that the model can be kept free of code that would freeze it to a particular input/output configuration.

In the following subsection the implementation of the Scrollable List (Name and UPT Number) (forward reference Section 5.6.1) shows the benefits of MVC architecture.

5.6.1 Example of MVC Architecture

The Scrollable List as illustrated in Figure 5.2 was implemented to facilitate the UPT user in using the interface. It contains a list of names and numbers of UPT users. Conceptually it is similar to the telephone number booklet which most of us are familiar with. The Scrollable List interface enables the UPT user to select a UPT number when communication is desired. In addition the UPT user can also edit the list by using the three buttons (Add, Delete, Modify) placed on top of the list. The operation of these three buttons has been described in Section 5.6.1. For instance, if I were to call Fergus on the phone, then, all I need to do is click on the line representing Fergus. The name Fergus O'Brien and his UPT number now appear against the *Name* and *Number* fields. I now need to click on the *Phone* button to initiate the call.

The MVC architecture was used in implementing the Scrollable List. Take a look at the illustration in Figure 5.3. Now we have an additional name (Anoop) and UPT number (9115501002) pair in the Scrollable List. The highlighted line across the name and UPT number pair indicates that it is currently selected. The user can now either delete this pair from the list or modify it. Let us assume that the user wants to delete it from the list. To do this the user must click on the *Delete* button. *Anoop 9115501002* disappears from the list, and the resulting Scrollable List is as illustrated in Figure 5.2.

When the *Delete* button was clicked, a message got passed to the model object which removed *Anoop 9115501002* from the list. In this case the list of names and UPT numbers is part of the user profile. When the *model* changes, then due to the dependency mechanism offered in Smalltalk, the *view* redisplays itself. The resulting list after the delete operation is as illustrated in Figure 5.3.

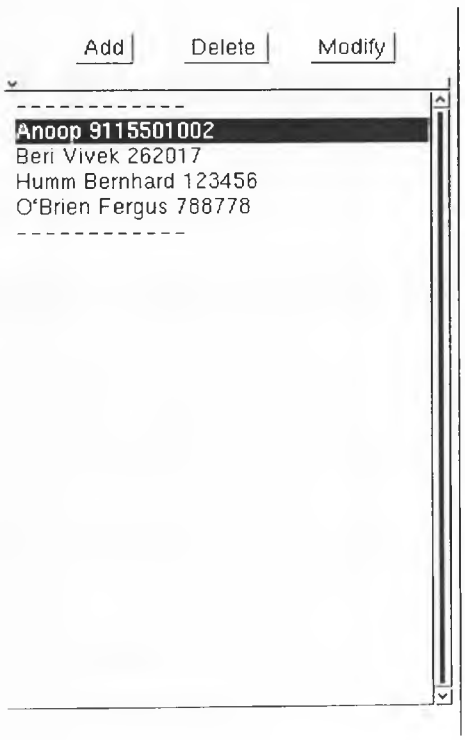

Figure 5.3: Scrollable List

Chapter 6

Design and Implementation

6.1 What Should Our Interface Look Like?

It was envisaged to develop a prototype for the user interface for the advanced telecommunications services (defined in Chapter 3) with multi-media communication. The user interface to the multi-media telecommunication ought to be easy to learn and use. An attempt was made to incorporate most of the user interface design principles and issues discussed in the previous chapters. The interface in it's initial state is illustrated in Figure 6.1.

6.2 Scope of the Prototype

The prototype of the user interface for the advanced telecommunications services covers interfaces for all the modes of communication listed in the next section. In addition to these, other interfaces such as the Scrollable List (discussed in Section 5.6.1) has also been developed. The *diary* interface was also developed which is discussed in Section 6.8. As this is only a prototype it was important to

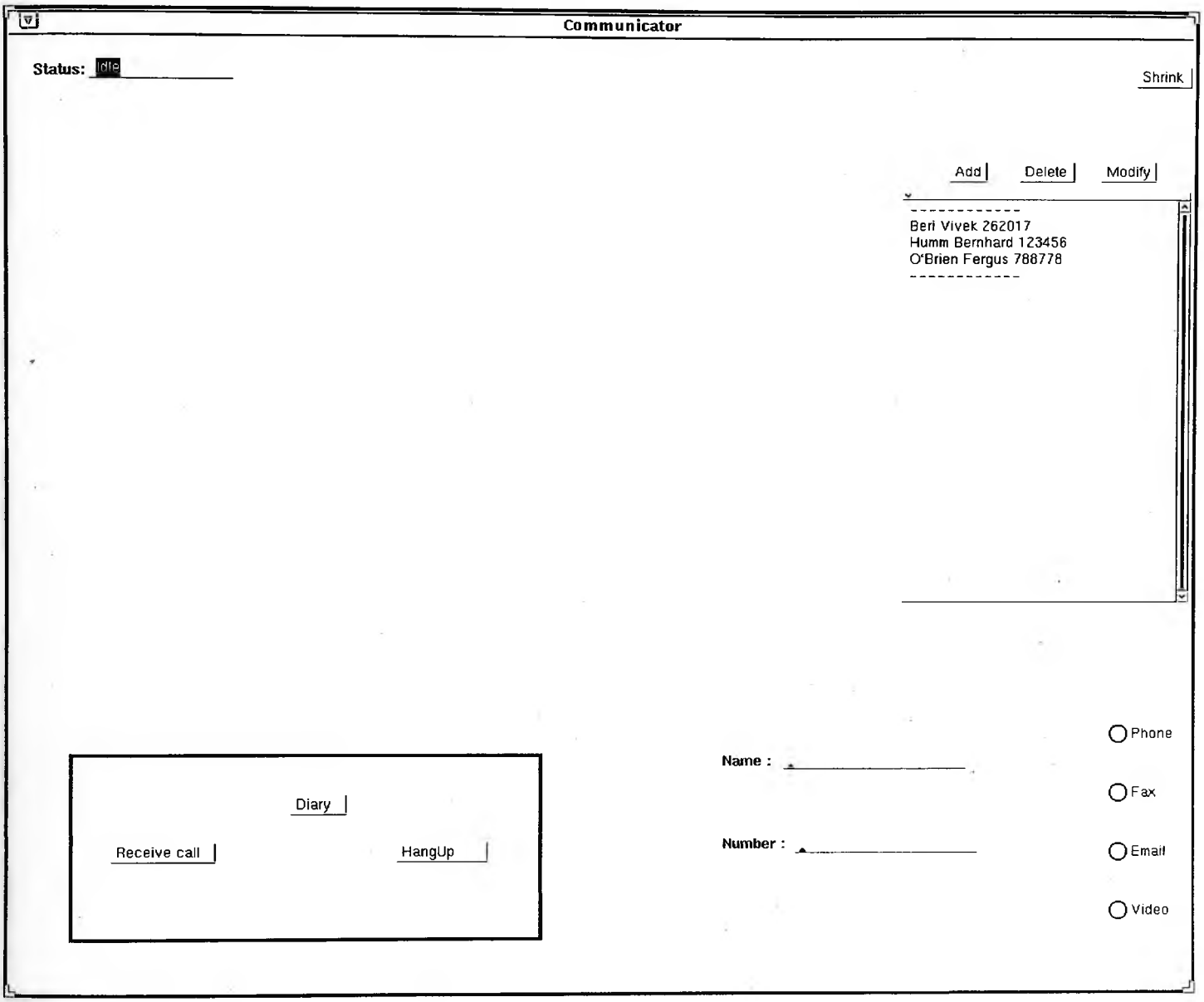

Figure 6.1: UPT User Interface

get the general "look-and-feel" of the interface rather than trying to incorporate all the possible functionalities.

However, certain things were taken for granted while developing the prototype. It was assumed that workstations supporting multi-media were available, and the underlying communication network existed. Microphones, audio speakers, video camera, and facsimile machine having hardware connections with the workstations exist. The network for communications also exists, supporting transmission requiring high bandwidth (real time video transmission).

6.3 Interface Components

The design process at its most rudimentary stages involved breaking up into different media which were to be considered. We isolated the following modes of telecommunication.

- Phone (common telephone)
- Facsimile
- Electronic mail
- Video conferencing

Each of these has been discussed individually in the sections ahead.

6.4 Phone Interface

The phone interface is as shown in Fig. 6.2. In order to make a phone call the user does the following.

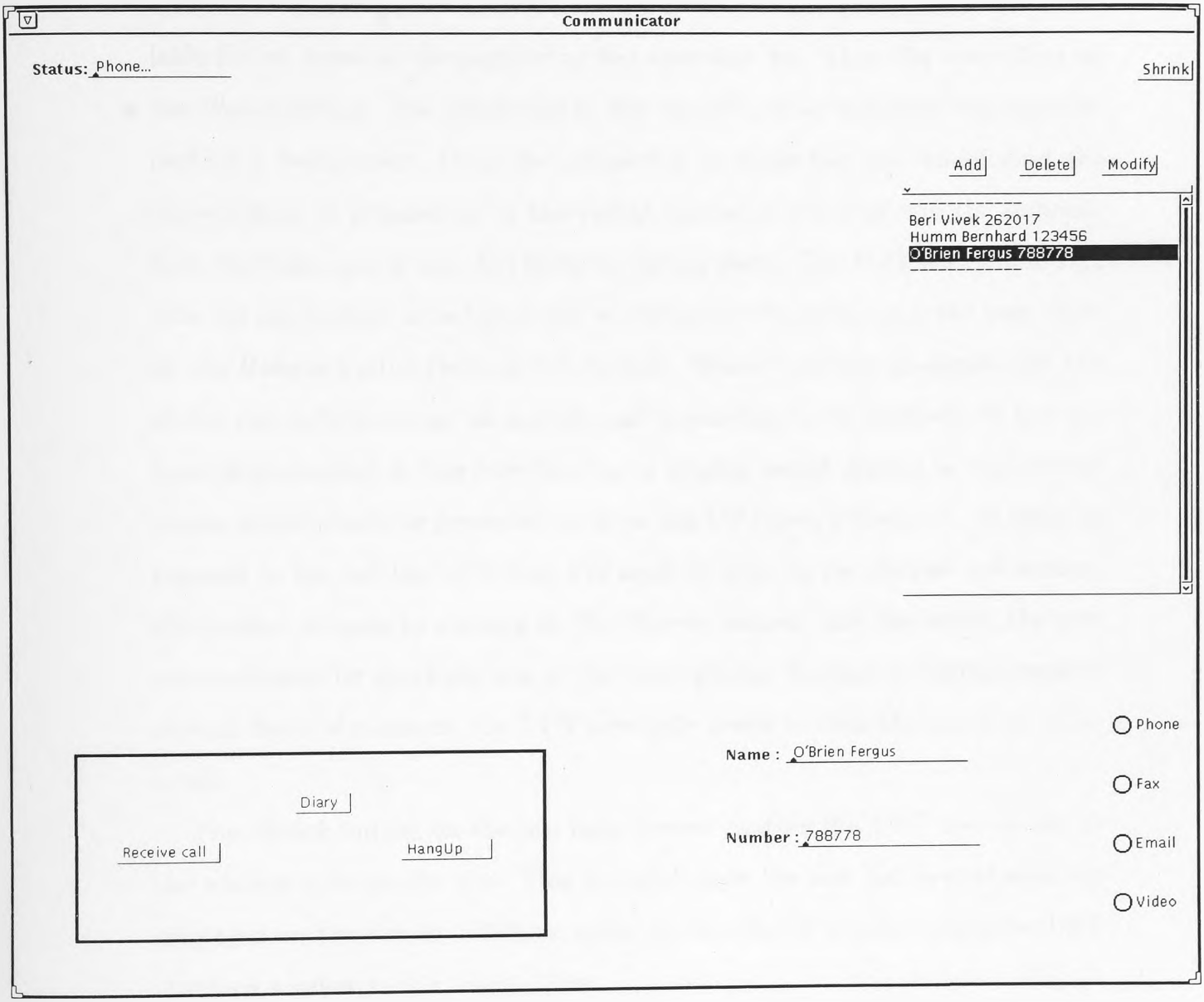

Figure 6.2: User interface

The UPT user selects a UPT number (of the called party) from the scrollable list or types in the number as the case may be. Then the user clicks on the *Phone* button. The status line in the top-left corner indicates that the connection is being tried. Once the connection is made the user would start the conversation by responding to the verbal salutation (or whatever the response from the called party may be) from the called party. The UPT user would talk into the microphone attached to the workstation. To disconnect, the user clicks on the *Hangup* button (bottom-left corner). When receiving an outside call, the status line indicates that an outside call is pending to be received. It has not been implemented in this interface but a ringing sound similar to the current phone system could be generated to draw the UPT user's attention. In order to respond to the call the UPT user will need to click on the *Receive call* button. Connection is made by clicking on the *Receive* button, and like before, the user communicates by speaking into to the microphone. Instead of dialing/pressing several digits of numbers, the UPT user only needs to click the mouse to make a call.

The *Shrink* button on the top right corner enables the UPT user to shrink the window to a smaller size. This is useful when the user has several windows displayed on the screen. Clicking again on the *Shrink* button opens the UPT interface window to it's original size.

6.4.1 Evaluation

The current advanced phone systems have been criticised by Dr. Donald A. Norman. Though these systems offer many functionalities and features, the *mapping* [Nor88a] problem is severe. Dr. Norman discusses mapping problems in

advanced telephone features, but here we will deal with relatively simpler issues. Most modern telephone instruments these days have function keys. The most common use of function keys is for storing telephone numbers of people the user may be calling frequently. Use of function keys reduces the number dialing process to pressing a single key, but at the same time the instrument user is forced to remember which function key holds which number. The user is quite likely to make a slip and press the wrong function key (the fact that they are all arranged so close together). There is no way of detecting this slip before it is discovered by actually speaking to the called person at the other end. In addition, this feature is instrument dependent.

In our phone interface, the UPT user need not remember any UPT number. Instead of storing UPT numbers in function keys (there are no function keys), the UPT user has the option of storing them in Scrollable List. These stored UPT numbers form a part of the user profile which the user can modify at any time. The Scrollable List has been described in Section 5.6.1. The UPT number is selected by clicking on the line representing the UPT number. As a feedback of the action just performed, the name and UPT number selected get displayed in the *Name* and *Number* fields. The user has a chance to view the selection and change it if it happens to be an incorrect one. Lastly, under UPT, telecommunications services are not device dependent (refer Section 3.1).

6.5 Facsimile Interface

The user gets into facsimile mode by clicking on the *Fax* button. The interface we have provided here consists of features that are common in several facsimile machines. The interface is as illustrated in Fig. 6.3. The design of the

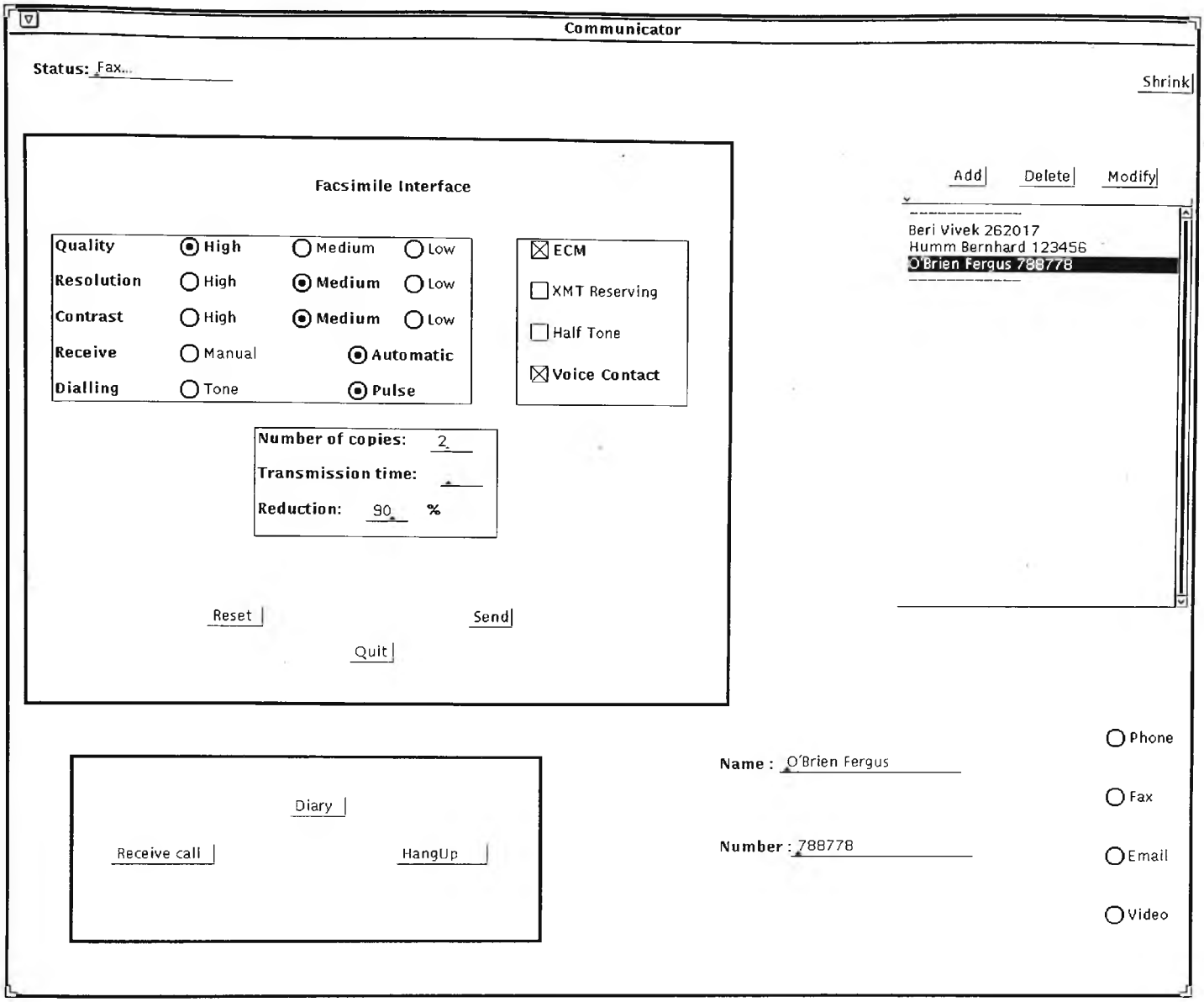

Figure 6.3: Facsmile interface

Fax Interface was influenced by the Macintosh Print Menu in Microsoft Word. Mutual exclusion was provided making the interface systematic. Buttons with related functionalities were grouped together. Parameters having any one of the three values (Quality, Resolution, etc.) are grouped together (first group), toggle switch parameters (ECM, Half Tone, etc.) are together in a box in the right (second group), and parameters that require keyboard entry (Number of copies, Transmission time, etc.) are grouped together (third group).

Illustration 6.3 shows that the UPT user sets parameters in the first group as

follows; *Quality* to *High*, *Resolution* to *Medium*, *Contrast* to *Medium*, *Receive* to *Automatic*, and *Dialing* to *Pulse*. In the second group *ECM*, and *Voice Contact* are set. From the third group it is evident that a 90% reduction will be applied to the two copies transmitted. The values for parameters in group one are mutually exclusive. That is, if the user selects *High* then a black dot appears in the *High* switch indicating that it has been selected. Now if the user selects *Low* for the same parameter then the black dot shifts to the *Low* switch. At one time only one of the three values can be selected for any parameter in group one. As for parameters in group two, they can be in either ON (selected) or OFF (deselected) state.

If the UPT user clicks on the *Reset* button, then all the parameters in all the three groups get initialized. By clicking on the *Quit* button the user terminates the facsimile interface. After setting all the required parameters the user completes the action of sending a facsimile document by clicking on the *Send* button. This document is a file containing graphics, or text, or a combination of the two. If a typed document is to be faxed, then it will need to be scanned through a scanner first. At the receiving end there may only be a conventional fax machine, or the UPT user at the receiving end may have disabled all communication in his/her diary (Section 6.8) except for fax. In this case the document will be delivered as a normal fax message. However, if the receiver is open to email, then the same document could be delivered via email. To specify the document, the UPT user enters its filename in the *Filename* field. The document gets delivered at the selected UPT number which is specified in the *Number* field (in this case it is 788778).

6.5.1 Evaluation

We will use Dr. Norman's design questions to evaluate the facsimile interface.

How easily can the user determine the function of the device? The user clicks on the *Fax* button when he/she wishes to communicate with another person through a fax document. The UPT user is already aware of the function of the device, in this case a fax machine.

How easily can the user tell what actions are possible? All the parameters have been classified into three groups. In group one there are three values for each parameter. The names of the values are *High*, *Medium*, and *Low*, which indicate that a selection of any one is required from the three available choices.

How easily can the user determine the mapping from intention to physical movement? The only possible way the user can select a value for a parameter in group one and two is by clicking on it. The moment the user does this, a black dot appears inside the selected switch indicating that it has been selected. The system is easy to learn. The availability of the *Reset* button (like the UNDO command) gives the user confidence to experiment. The user is not scared of making mistakes because the initial state can always be reached. A experimentation aids the user in determining the mapping from intention to physical movement.

How easily can the user perform the action? The action of selecting a value for a parameter is easily performed by a single mouse click. To reset the user needs to only click on the *Reset* button.

How easily can the user tell what state the system is in? The current state of the system as a whole is indicated on the *Status* line in the top left corner. Black dots inside the switches also act as status indicators. The user can directly manipulate the objects.

How easily can the user determine the mapping from system state to interpretation? By looking at the *Status* line the UPT user can easily tell what state the system is currently in.

How easily can the user tell if the system is in the desired state? The settings in this interface are an example of WYSIWYG (what you see is what you get). At any point in time the interface shows the state of the system, that is, it shows which parameters have been set and which have not. The user has a clear understanding of the state of the system. If the user is happy with the current settings then he/she acts to have the fax delivered by clicking on the *Send* button.

It was felt that sometimes it is more convenient to input using the keyboard. For example, in the case of *Number of copies*, and *Reduction* it would be probably easier for the user to enter 2 and 90 respectively. Entering from a keyboard in this case is easier than with screen selection. The keyboard remains the most effective direct manipulation device for some tasks [Sch87].

6.6 Email Interface

The email interface looks like as shown in Fig. 6.4. When the user clicks on the *Email* button the Email Interface comes up while the base window remains unchanged.

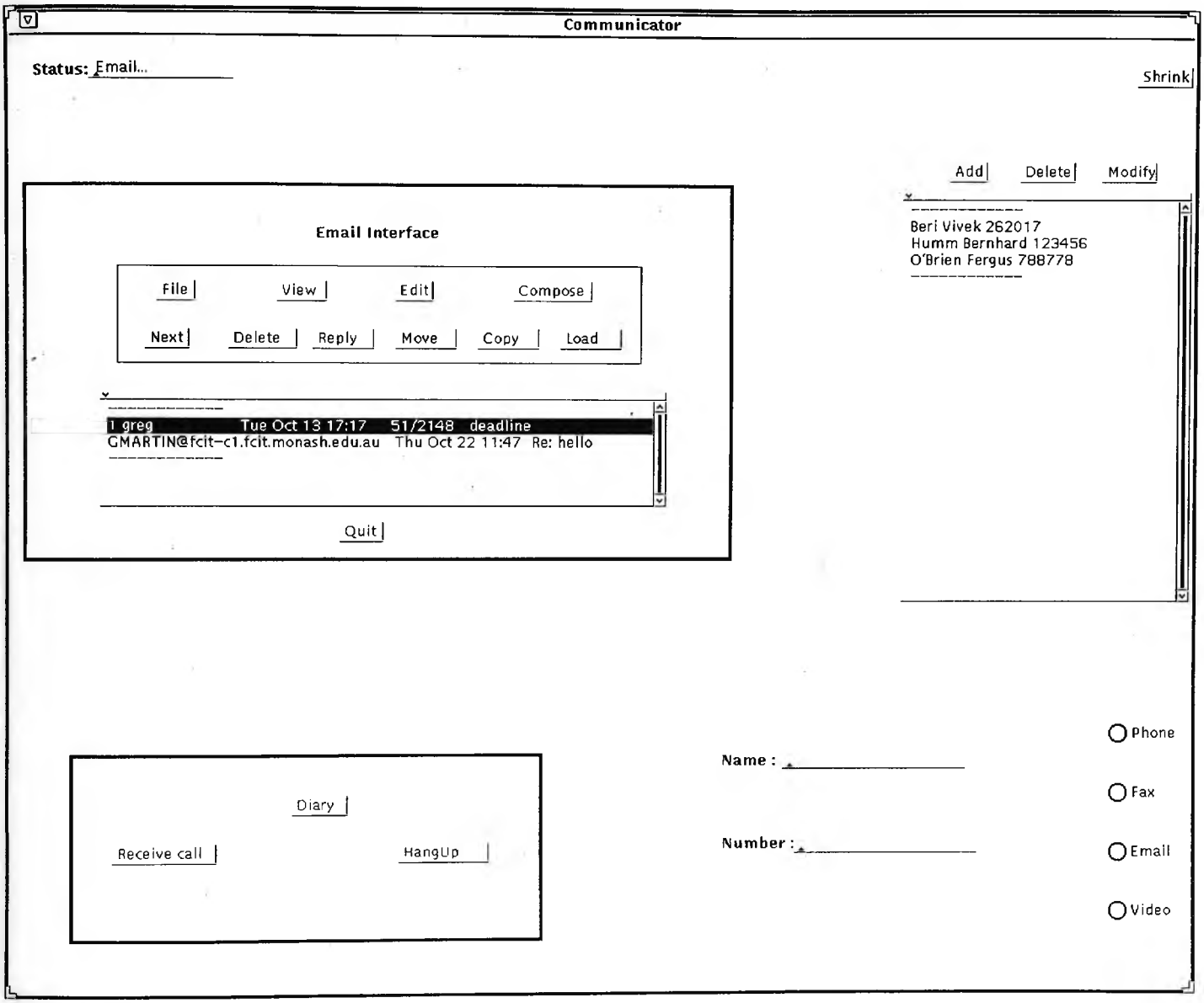

Figure 6.4: Email interface

Our email interface offers several buttons to the user for going about the tasks of sending and receiving electronic mail. The user needs to enter the UPT number in the *Number* field and click on the *Email* button. In doing so the edit window shows up allowing the user to compose a message. The user can also reply to the sender of received messages. This is done by selecting any of the received messages from the scrolling window and then by clicking on the *Reply* button. A reply could be sent to the sender of the message, or to all. The user also has the option of including the original message in his/her reply.

6.6.1 Email Interface Components

The Email Interface has three separate components. Firstly, there is a rectangular box containing ten buttons. Secondly, there is a scrolling window which contains the headers of received mail, and lastly, there is a *Quit* button. The *Quit* button terminates the email session and the Email Interface window disappears. In functionality the UPT Email Interface is similar to the one provided by Sun. For description of all the buttons, the reader is referred to the Desktop SPARC, Sun Systems User's Guide [Sun91]. However, the Deliver functionality in the SPARC desktop is not well designed and is evaluated in the following subsection.

6.6.2 Evaluation

After composing a message the user's intention now is to have the message delivered. In the SPARC desktop Mail Tool, a *Deliver* button is provided. First time round when the user sees this button and wants the composed message to be delivered, he/she clicks on it. On doing so the compose window disappears

without any indication as to what happened. If the user explores the choices offered by the *Deliver* button, then a pop-up menu appears with four options which are:

1. Quit window
2. Close window
3. Clear message
4. Leave message intact

These options take the user away in thought from the initial intention of delivering the message. Here the focus has shifted to the window. It is not clear that on selecting *Quit window* the message will be delivered. Users learn the workings of this interface but only after stumbling over it once. It may cause some dissatisfaction among users who are already familiar with ELM and are used to the "Mail Sent!" message.

In the UPT interface design we have tried to rectify this by providing a *Status* line on the top left hand corner of our interface. This acts as a good measure for the user who needs to know the status of the system from time to time. When the *Deliver* button is clicked, the *Status* line indicates that the message has been delivered.

Also, we will use Dr. Norman's design questions to evaluate the remaining functionality of the email interface.

How easily can the user determine the function of the device? The user clicks on the *Email* button when he/she wishes to communicate using the electronic mail system. The UPT user with moderate familiarity with computer

systems would be aware of the function of the device, in this case an email system.

How easily can the user tell what actions are possible? All the functionality has been classified into three separate components. In the first component there are ten buttons. The names on these buttons serve as a good indication to the function they perform. The second component contains a scrolling selection list. Such a list is used for selecting an item and its functionality is straight forward. The third component contains a single push button *Quit*. Clicking on this terminates the email session and hence the action it carries out is pretty clear.

How easily can the user determine the mapping from intention to physical movement? The only possible physical user interaction at this stage is a mouse click from the user. For example, if the UPT user wishes to compose a new electronic message. The mapping from the intention (want to compose a message) to the physical movement is straight forward and the user only needs to click on the *Compose* button. Similarly, the other buttons provide a simple mapping from intention to physical movement.

How easily can the user perform the action? The only action that the user can perform is a mouse click and that is easily achieved. However, at a later stage the user may be required to type in an entire message which may involve the working knowledge of a text editor. That too comprises only of keystrokes and mouse clicks.

How easily can the user tell what state the system is in? The current state of the system as a whole is indicated on the *Status* line in the top left corner.

How easily can the user determine the mapping from system state to interpretation? By looking at the *Status* line the UPT user can easily tell what state the system is currently in.

How easily can the user tell if the system is in the desired state? At any point in time the interface shows the state of the system, that is, for example, if the user is composing a message then the active editor would indicate that the user is in editing/composing state. The user has a clear understanding of the state of the system.

6.7 Video Interface

The video interface looks like as shown in Figure 6.5. In this thesis we have restricted our interface design to support only a single person video conferencing facility.

When the user clicks on the *Video* button (bottom right corner) the Video Interface and Video Image as illustrated in Figure 6.5 comes up. We are not concerned with the underlying network and assume that the connection has been made (with 793256787 in this case). The remote person's picture (or whatever the camera is aimed at) appears in our Video Image window. The Video Image window can be dragged and placed anywhere on the screen.

The Video Interface, as illustrated in Fig. 6.5, has seven buttons. Five of these are similar in functionality to the buttons on a VCR. These are *Rev*, *Play*, *Fwd*, *Stop*, and *Rec* buttons. The *Quit* button enables the user to terminate the video session. The *Quit* button is provided in Fax Interface, Email Interface, and Video Interface and has the same function in all. The functionality of the

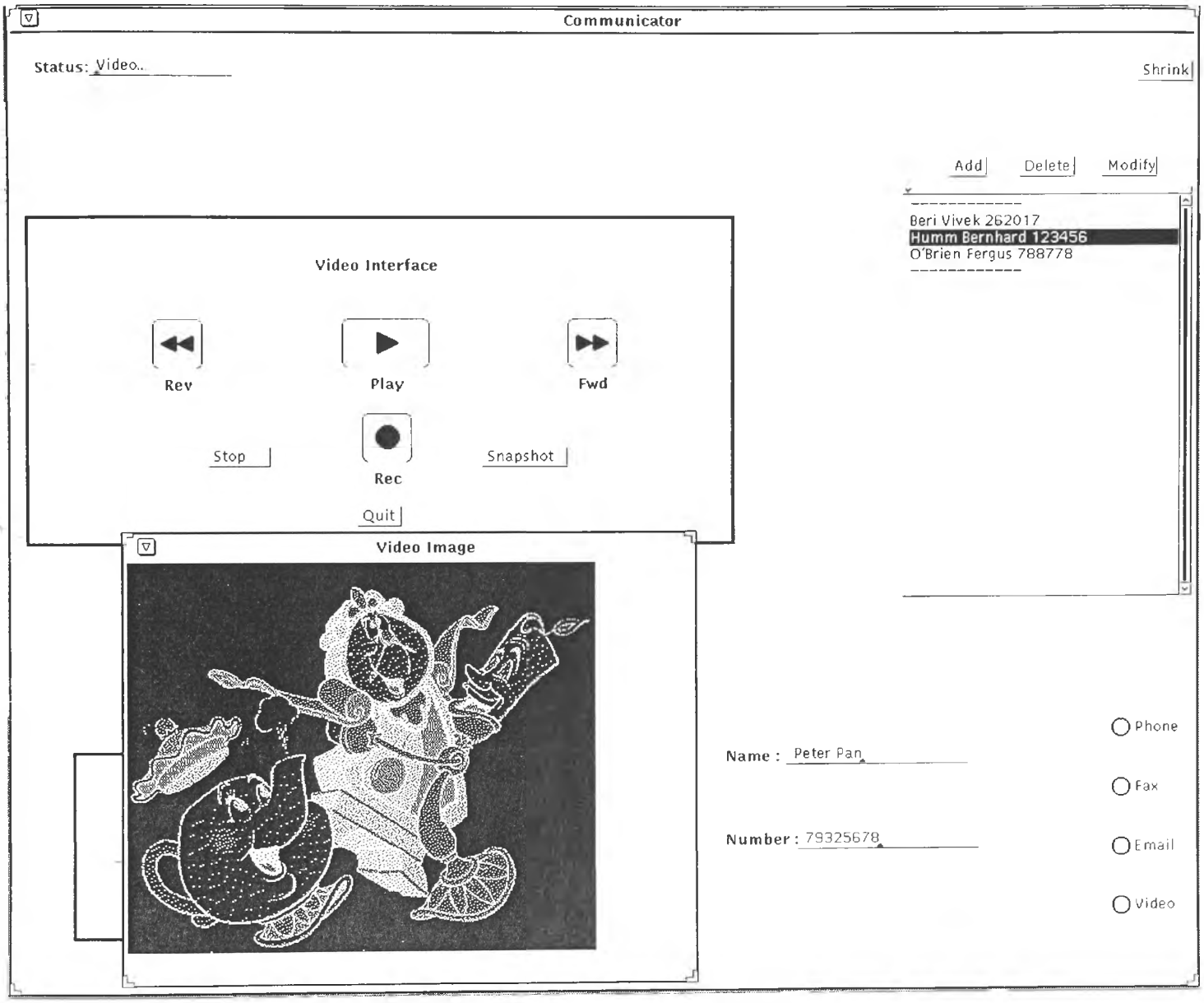

Figure 6.5: Video interface

buttons is briefly touched upon as follows.

1. **Snapshot:** During a video session, a user can take a snapshot of a portion of the image being displayed in the Video Image window. By clicking on the *Snapshot* button the cursor on the screen changes shape prompting the user to specify a region of the image. The user has to click on the mouse, keep the mouse button pressed, and drag the mouse over the region to be captured. Marked lines indicate the region already selected and offer a feedback to the user. For the snapshot to be taken the user has only to lift his/her finger from the mouse button. The user would then be prompted to specify a name for the file to hold the snapshot.

2. **Rev:** Moves back in a video recording. To stop rewinding, the user needs to click on the *Stop* button.

3. **Play:** When the user clicks on the *Play* button, a pop-up menu containing a list of existing video recordings appears. To view a recording the user simply selects one from the list.

4. **Fwd:** Moves forward in a video recording. To stop fast forwarding, the user needs to click on the *Stop* button.

5. **Stop:** By clicking on the *Stop* button, the user is able to stop rewinding, fast forwarding, or playing of a recording. This is similar to the stop button on a VCR.

6. **Rec:** The user may wish to record the entire video conference or only a part of it for the benefit of reviewing it at a later time. To start recording the user has to click on the *Rec* button. To stop recording the user clicks on the *Stop* button.

The image would be recorded/stored on hard disks and unlike the VCR there is no tape involved. However, it was felt that using similar buttons would provide a good metaphor. The functionality is the same and the user does not care whether the storage medium is tape or disk.

The benefits of the above mentioned media in telecommunication are well accepted and have been omitted here.

6.7.1 Evaluation

Using Dr. Norman's design questions, the UPT Video Interface is evaluated as follows.

How easily can the user determine the function of the device? The user can easily determine the function of the device as he/she views the interface. The name *VCR*, of the button indicates it's functionality.

How easily can the user tell what actions are possible? The possible actions are specified by the buttons provided in the Video Interface. Functionality of all of these buttons is social general knowledge. Most UPT users would have seen similar named buttons on VCR's and cassette recorders. Some of the buttons are graphic icons which makes their functionality more clear.

How easily can the user determine the mapping from intention to physical movement? Consider for a moment that the UPT user wishes to record part of a video conference. The thing that comes to the users mind first, is that it has to do something with the *Rec* button. The only possible user's operation on a button is to click on it. The only way the user can

do this is by using the mouse (constraints are applied here to make a simple mapping). Hence, the user clearly understands the mapping from intention (record) to physical movement.

How easily can the user perform the action? Considering the same example of recording, the UPT user can easily perform the action (click on the *Rec* button) using the mouse.

How easily can the user tell what state the system is in? The state of the system is displayed on the *Status* line. For example, if the UPT user clicks on the *Fwd* button, then the *Status* line indicates, "Fast Forwarding...".

How easily can the user determine the mapping from system state to interpretation? The textual displays on the *Status* line are descriptive in nature and indicate the exact state the system is currently in.

How easily can the user tell if the system is in the desired state? The objects and actions are visible to the user. At any point the user is aware of the direction towards the goal and can easily change if it happens to be an undesirable one. Tasks (functionalities offered) are shallow (Section 2.8) in nature and the user remains in full control over the system.

6.8 Diary Interface

The UPT diary is conceptually similar to the paper diary with which probably all of us are familiar. Both are useful devices assisting the human memory. However, the UPT diary has more specific utilities, it assists the UPT user to communicate *asynchronously*, in a *time independent* fashion. The UPT user will

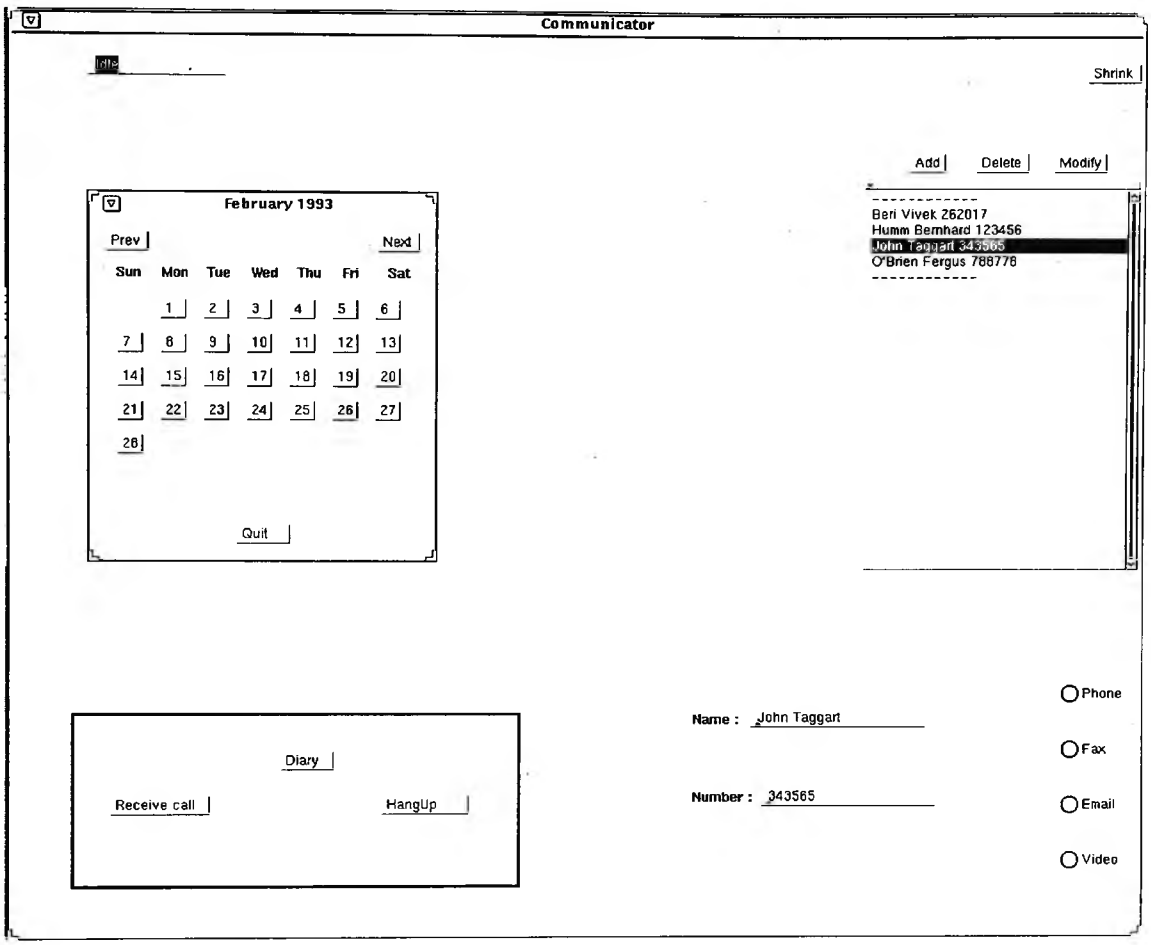

Figure 6.6: Diary Calendar Interface.

be able to consult the diary of the called party at the time he/she is updating his/her own diary.

The most common use of the paper diary is to note an appointment. To do this, first of all we open the diary to the page showing the date of appointment. One may then write down the time of appointment and put a comment in the space provided. The UPT diary interface is somewhat similar. The engagement information (time and reason) as well as the communication media are noted. In the following paragraphs, I have described the working of the UPT diary

with the help of examples and illustrations.

The UPT user clicks (using a mouse) on the *Diary* button on the main interface screen to bring up the *calendar* interface as shown in Figure 6.6. A new window for the calendar interface for the current month gets displayed. Using the *Prev*, and *Next* buttons on this window, the UPT user can get the calendar displays for other months. The dates of each month act as buttons, by clicking on them the user can open any page of the diary. For example, if the UPT user clicked on the button labeled *26*, then, a new window depicting the UPT diary page appears as illustrated in Figure 6.7. This page can be broadly classified into three main columns and also has two control buttons at the bottom of the window. The first and second columns represent time slots and the comment space respectively. This conforms to the layout of a page of a normal paper diary. However, the UPT diary has an additional column as well. This third column consists of f our check boxes for each of the time slots. They are labeled as *Phone*, *Email*, *Fax*, and *Video* respectively. These check boxes can be set (communication enabled) or reset (communication disabled) by clicking on them. As is illustrated in Figure 6.6, the initial state of all *Phone*, *Email*, and *Fax* check boxes is set. The *Video* check boxes are reset.

Let us assume for a moment that John Taggart is an UPT user and has opened his diary page for Friday, 26 February 1993 for update. John has a meeting with Gary from 9am to 10am and types a note in the comment space provided. John also does not want to be disturbed during the meeting so disables (reset) the *Phone*. As he would be at the meeting he is not available for video conferencing and hence John disables the *Video* check box too. The same procedure is followed for the 2-3pm appointment and the *Phone* and *Video* at that time are also disabled. During these two hours (9-10am and 2-3pm) John

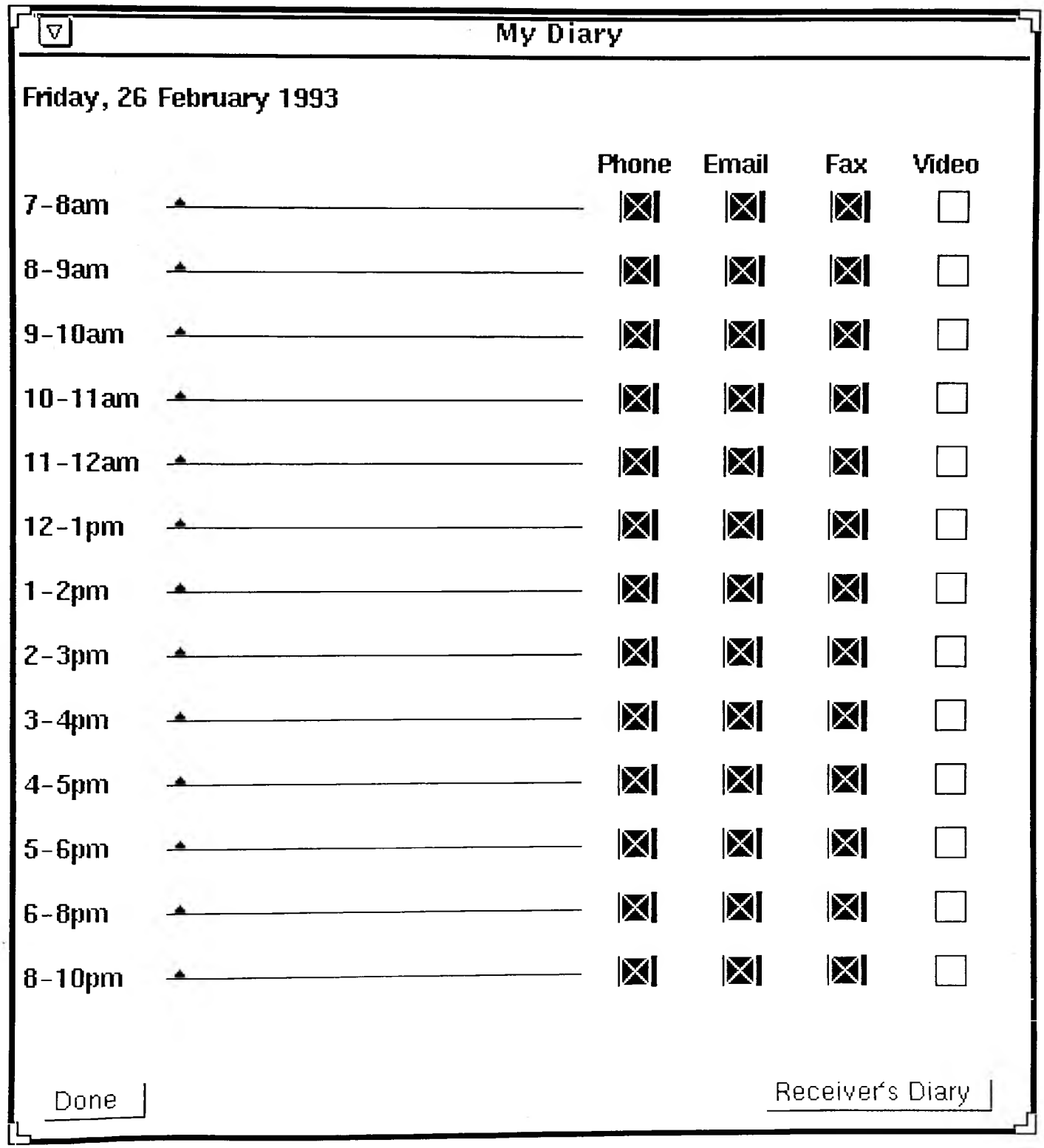

Figure 6.7: UPT User's Default Diary Page Interface.

is open to communication through email and facsimile only. Assuming that these were the only appointments for the day, John has now updated his diary and any UPT caller will consult John's diary before communicating with him. John's diary page for Friday, 26 February 1993 now looks as shown in Figure 6.8.

For example, say a UPT caller wishes to communicate with John Taggart on Friday, 26 February 1993. The user already has his/her diary page open and clicks on the *Receivers Diary* button. On doing this John Taggart's diary page for the specified date gets displayed as illustrated in Figure 6.9. Due to privacy issues the comment column from John Taggart's diary have been omitted in the display. All that John wants to communicate through his diary is that he is unavailable on phone and video for those two hours, what he would be doing at that particular time is personal to John. After viewing John's diary for an arbitrary day the caller is aware of John's availability/unavailability for communication on different media. For instance, let us assume that it is 9:30am and somebody wishes to talk to John on the phone. The caller consults John's diary and sees that John does not wish to communicate through the phone. However, the caller now also knows that John will be available on phone after 10am until 2pm and then from 3pm to 10pm. It is up to the caller to talk to John during this period or to communicate using alternative media (John is always available on email and facsimile).

It is expected that UPT users will consult the diary before communicating (specially prior to using the phone and video) with other users. By disabling a media for a period of time the UPT user indicates his/her preference for mode of communication. For example, John may be having a meeting in his own office. He could have the phone disconnected (no calls will be received) or leave

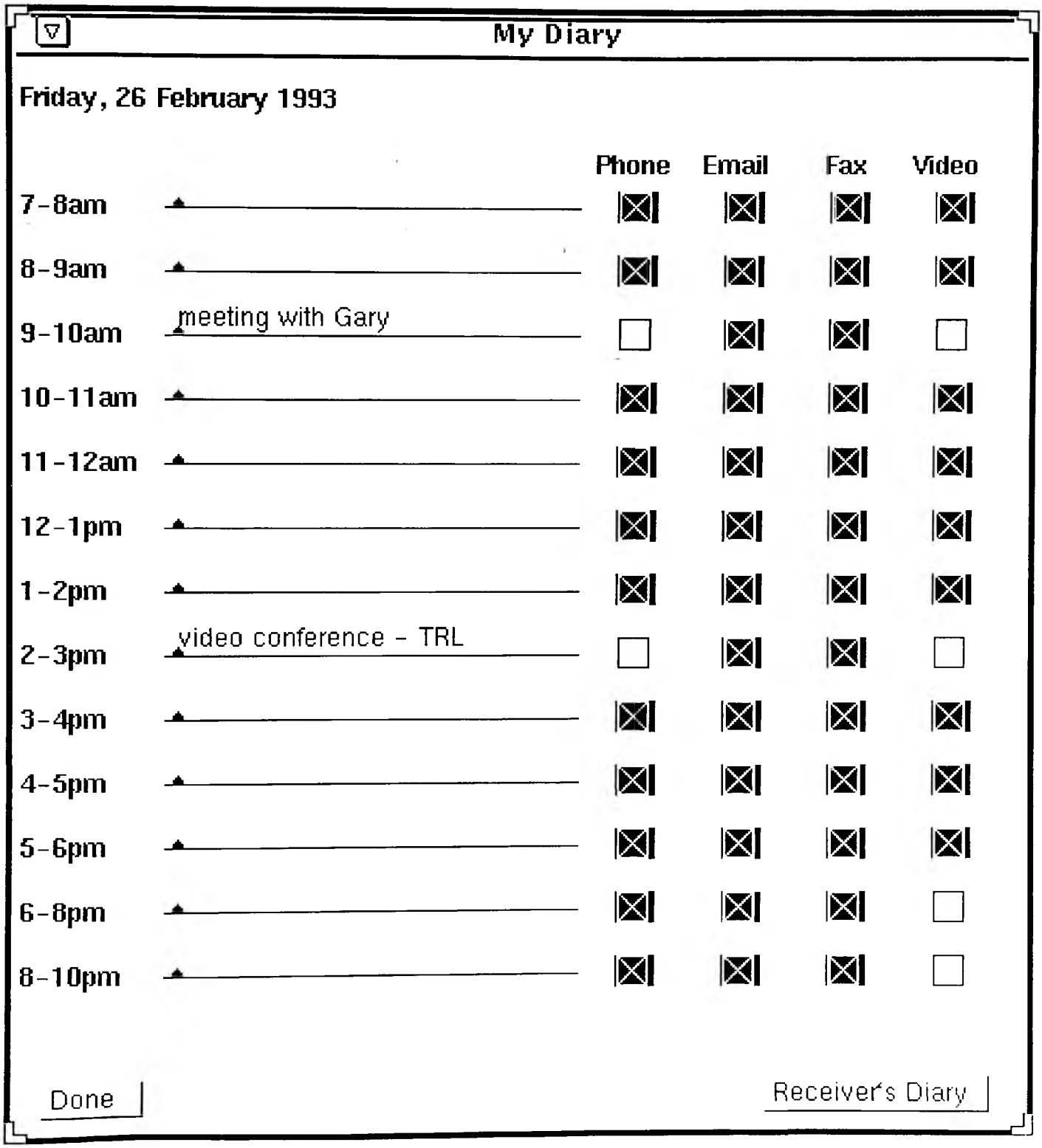

Figure 6.8: UPT User's Diary Page.

it as it is but just disable it in his diary. If he has the phone disconnected then what about emergency calls? If he leaves it as it is then the phone will ring if someone calls and John can only hope that it is something which needs his immediate attention. These are issues which need to be addressed but are outside the scope of this thesis.

At any time an UPT user can be consulting diary pages of several UPT users, or different diary pages of the same user, or a combination of the two. When the UPT user clicks on the *Receivers Diary* button (illustrated in Figure 6.8) the diary page of the UPT user whose name and number are specified in the *Name* and *Number* fields on the main interface screen (Figure 6.6) gets displayed. To consult diaries of different UPT users the user has to change the contents of these two fields. The *Name* and *Number* fields have been discussed in relation to the *Scrollable List* in Section 5.6.1.

6.8.1 Evaluation

Using Dr. Norman's design questions, the UPT Diary Interface is evaluated as follows.

How easily can the user determine the function of the device? The goal that the user wishes to achieve is reached by breaking it up into several intermediate and comprehensible steps. For example, say the goal is to note an engagement in the diary for an arbitrary day. The user begins to satisfy this by first clicking on the *Diary* button. This is the only button (active object) relevant to the goal. On doing this action the window for the calendar pops-up. By looking at this calendar window it is quite easy for the user to determine its function (selecting a day) with respect to the goal.

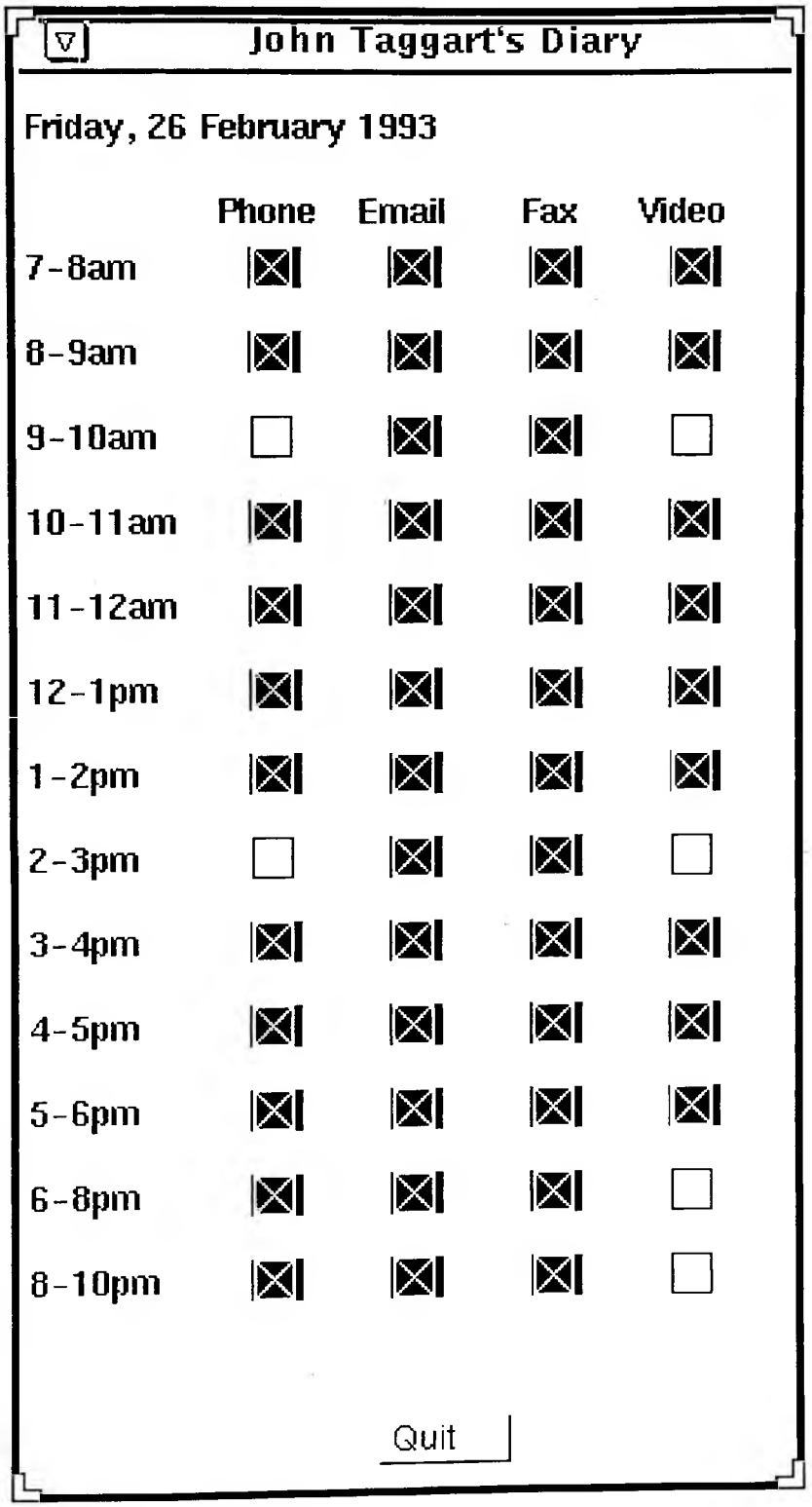

Figure 6.9: UPT Receivers Diary Page Interface.

This is the second step towards the goal. As the user clicks on any one of the days of the displayed month, the diary page pops-up. The date on the page confirms that it is indeed the correct page. The user now notes the engagement, the procedure for which is straightforward.

How easily can the user tell what actions are possible? Every interface has control buttons with self explanatory labels. For example, the Calendar Window has *Prev* and *Next*. It being a monthly display, it is not difficult for the user to understand what possible action these buttons provide. Similarly, the check boxes (shown in Figure 6.8) indicate their function.

How easily can the user determine the mapping from intention to physical movement? The mapping from intention to physical movement is easy to determine for these interfaces. For example, if the UPT user wishes to consult the diary page of another user then by identifying the *Receivers Diary* button on the interface the mapping from intention to physical movement is easily determinable.

How easily can the user perform the action? The action is easy to perform. Most of user's interaction is via mousing—point and click.

How easily can the user tell what state the system is in? The objects (check boxes and dialog boxes) are directly manipulatable. User's action is reflected in the visual display (WYSIWYG). This gives the user a good measure of the current state of the system.

How easily can the user determine the mapping from system state to interpretation? The mapping from system state to interpretation is a simple one. For example, clicking on the check boxes toggles their state from set to

reset and vice versa. The check boxes are displayed differently in either case and it is easy for the user to figure out the mapping.

How easily can the user tell if the system is in the desired state? At any point in time, the UPT user is aware of the state of the system. The look of the interface itself indicates that. By comparing the current state with the desired state, the user can tell whether the system is in the desired state or not.

Lastly, in all of the interfaces discussed, most of Ben Schneiderman's concerns about perception (Section 2.5) are satisfied.

Chapter 7

Conclusions

Designing a good user interface for an application is a complex task and even more so when the application itself is a complex one. Designers have struggled in the past to develop good user interfaces but there are still no standard guidelines that guarantee the manufacture of a good design and thus a good user interface. The UPT service has complex issues that need to be addressed and is keeping many researchers busy. My task was to explore the multimedia on the user's terminal which was accomplished, and is reflected in the design of the interfaces that were implemented.

A prototype of the user interface was successfully developed that caters to the four (phone, fax, email, video) modes of communication and provides an UPT user mobility in media. The bit mapped (window) screen plus voice would form the basis for a simple UPT-terminal device. On the basis of the user interfaces presented in this thesis, and their usability evaluation, it would not be wrong to say that the interfaces presented support the basic functionalities of the UPT service. The illustrations in Chapter 6 are screen snapshots of the interface and give an indication as to what the actual design looks like. The

proposed interfaces were evaluated against Dr. Norman's evaluation technique and passed with flying colours.

The UPT *Diary* concept was explored and an interface for it has been presented. This new concept clearly differentiates the UPT service from the existing telecommunications services in that it enables the UPT user to communicate in a time independent (negotiated) fashion.

I enjoyed working on this research project and would like to further my research by extending it onto a globally distributed level if I were to get an opportunity in the future. Also, there are several user interface issues that need to be addressed and which would add to the functionalities that are only limited by one's imagination.

Bibliography

[AC87] Inc. Apple Computer. *Apple Human Interface Guidelines:*. Apple Computer, Inc., 20525 Mariani Avenue, Cupertino, California 95014, 1987.

[AH88] Sueann Ambron and Kristina Hooper. *Interactive Multimedia.* Microsoft Press, 1988. Foreword by John Sculley.

[Boo90] Grady Booch. *Object Oriented Design with Applications.* Benjamin-Cummings, 1990.

[CCI] CCITT – International Telegraph and Telephone Consultative Committee. *Draft Recommendation F.851 - Universal Personal Telecommunications Service – Principles and Operational Provisions. Version 5*, Question 35/I. Study Group I (28 May - 7 June 1991 Meeting) edition.

[CY91] Peter Coad and Edward Yourdon. *Object-Oriented Analysis.* Yourdon Press, New Jersey, 2nd edition, 1991.

[GR89] Adele Goldberg and Dan Robson. *Smalltalk-80: The Language.* Addison-Wesley, 1989.

[Hat81] Don Hatfield. Easier and more productive use of computer systems. Lecture at Conference on Easier and More Productive Use of Computer Systems, 1981. Conference held at Ann Arbor, MI.

[HF92] Bernhard G. Humm and Michael Fazzolare. Object-Oriented Analysis for Telecommunications Services. In *Proceedings of the 1992 International ACM/SIGAPP Symposium on Applied Computing*, Kansas City MO, USA, March 1992.

[HHJ+86] Edwin L. Hutchins, Hollan, D. James, D. Norman, and A. Don. User centered system design: New perspectives on human-computer interaction. Lawrence Erlbaum Associates, 1986.

[Ing] D. Ingalls. The smalltalk-76 programming system design and implementation. In *Proceedings of the Fifth Annual ACM Symposium on Principles of Programming Languages*, page 9.

[Mul90] Mark Mullin. *Rapid Prototyping for Object-Oriented Systems.* Addison-Wesley Publishing Company, fourth edition, 1990.

[Nor88a] Donald A. Norman. *The Psychology of Everyday Things.* New York: Basic Books, 1988.

[Nor88b] Donald A. Norman. User interface strategies 88. Video, 1988. Workshop conducted at the University of Maryland, College Park.

[O'B91] Fergus O'Brien. Challenges for telecommunications engineers in meeting end-user needs. Melbourne, April 1991. Proc. First Australian Conf. Telecommunications Software - ACTS'91. Invited Paper.

[Row87] P. Rowe. *Design Thinking.* MIT Press, Cambridge, MA., 1987.

[Sch87] Ben Schneiderman. *Designing the User Interface: Strategies for Effective Human-Computer Interaction.* Addison-Wesley Publishing Company, The University of Maryland, May 1987.

[Sha85] B. Shackel. Ergonomics in design for usability. *HUSAT*, 1985.

[SM84] S. L. Smith and J. N. Mosier. The user interface to computer-based information systems: A survey of current software design practice. *Behaviour and Information Technology*, 3:195–203, 1984.

[Smi82] S. L. Smith. User-system interface. *Human Factors Society Bulletin*, 25(3), 1982.

[Sun91] Sun Microsystems, Inc., 2550 Garcia Avenue, Mountain View, California 94043-1100. *Desktop SPARC, Sun Systems User's Guide*, 1991.

[Thi82] Harold Thimbleby. What you see is what you have got? 1982. Unpublished paper, University of York, England.

[TK89] J. C. Thomas and W. A. Kellogg. Minimizing ecological gaps in interface design. *IEEE Software*, pages 78–86, January 1989.

[Tog92] Bruce Tognazzini. *TOG on Interface.* Apple Computer, Inc., 1992.

[TS85] J. C. Thomas and M. L. Schneider. Human factors in computer systems. *Human/Computer Interaction*, 1985.

[Wul80] W. Wulf. Trends in the design and implementation of programming languages. *IEEE Computer*, 13(1):15, 1980.